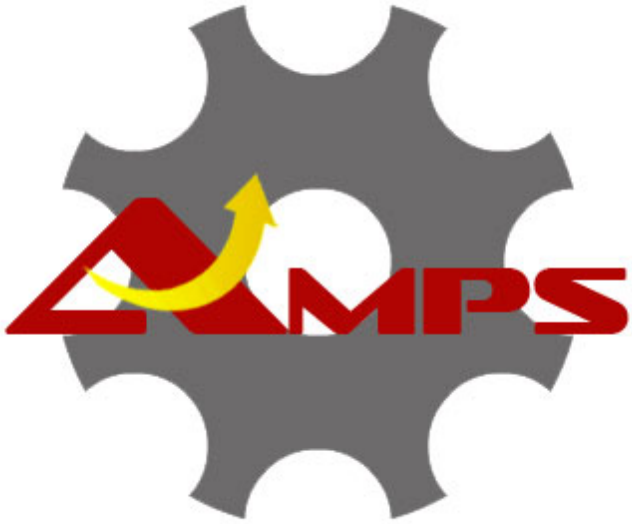
AUTONOMOUS MOBILE PORTER SYSTEM

FINAL REPORT
CMPE / EE 123B



11 JUNE 2011

CONTENTS

Team Members 4

Abstract 6

Motivation 6

Objective 7

Approach 7

- User Interface and Navigation 7
- Sensors 12
- Motor Control 19
- Microcontroller 22
- Feedback Control 24
- Power 28
- Structure 34

System Overview 40

Budget 41

Conclusion 43

Future Applications 44

Works Cited 48

Appendix A: Graphical User INterface 49

Appendix B: Electrical schmatics 53

- Sonar Sensors 53
- Power Rails 56
- Motor System 58

Bump System 60
Encoders 63
Appendix C: Mechanical Prototypes 64
Appendix C: Team Charter 69
Contact information 69
Overview 70
Code of Conduct 72
Agreement 75

TEAM MEMBERS

BRADY BOONE

Undergraduate Bioelectronics and Computer Engineer - University of California, Santa Cruz

I am a 3rd year transfer student majoring in Bioelectronics, Computer Engineering with a focus in Robotics, and a minor in Electrical Engineering. My engineering interests are in the applications of biological systems to engineering design of robotic systems. When I graduate in the Spring I would love to design robotic systems that assist disabled people, such as prosthetic devices. For this project I will be the lead engineer of mechanical and structural components. I will also be assisting with the design and implementation of the sensors.

ALEJANDRO CERDA

Undergraduate Computer Scientist - University of California, Santa Cruz

I am a 5th year Computer Science major with a minor in Computer Engineering. I consider myself to be a well-rounded student, having interests in robotics and software test. I have gained industry experience as an intern with Lockheed Martin Information Systems and Global Services performing software testing on satellite flight software. I will be continuing with the company upon graduation as an Associate Software Engineer. I enjoy taking on personal side projects as a hobby to build functional but really unnecessary gadgets like my recent dual arcade controller assembly made from an old computer keyboard. In my spare time I also like to constantly upgrade (or break) my computers at home, learning more and more about Windows and Linux features as I go. I will primarily be responsible for overseeing the user interface and software implementation for this project.

NOLAN LAU

Undergraduate Computer Engineer - University of California, Santa Cruz

I am a Computer Engineering student here at UCSC with a primary focus in the digital hardware track. I chose AMPS as my project because of the level of difficulty that is required for hardware. I was mainly intrigued by the sensors that I will be interfacing to a central processor. I will be graduating this Spring quarter with a Bachelors of Science in Computer Engineering and a minor in Electrical Engineering. Upon graduation, I plan to look for a job which lies in my field of interest, namely hardware design. My primary responsibility on this project is to lead the sensors hardware portion of the project.

PAUL NAUD

Undergraduate Electrical Engineer - University of California, Santa Cruz

I am 5th year Electrical Engineering major with a primary focus on electronics. I currently work for CITRIS at UCSC where I have gained strength in my research abilities and project management. I will be graduating in the Spring of 2011 and continuing my student career at UCSC as a graduate student of Electrical Engineering under Professor Patrick Mantey. My primary directive on this project is power distribution and communications.

CHASEN PETERS

Undergraduate Electrical Engineer - University of California, Santa Cruz

I am currently a 5th year Electrical Engineering student. A majority of my interest lay within hardware design. I have some previous experience working as an electrician assistant at Central Contra Costa Sanitary District, where I worked on their ultra violet disinfection system. I will be attending Stanford University next Fall to obtain my Masters in Electrical Engineering. I will serve as this project's team lead as well as being responsible for the motor control and on-board system interface.

ABSTRACT

Our goal is to effectively design and build an autonomous vehicle capable of intelligently delivering payload to pre-determined locations within a set structure. A vehicle like this would allow an employee's time to be allocated towards tasks, which are of higher importance in comparison to delivery services. Types of transported goods could range from mail and small packages all the way up to adult individuals. In regards to the transportation of individuals, this vehicle would be a means of assisting disabled people to an office or classroom of choice in the event that the environment is unfamiliar territory. The design is based on the modification of an existing powered wheelchair so that it may successfully direct itself throughout a building by using an array of sensors. Also included in the design is the ability to remove the chair and replace it with a locked storage box. This storage box would allow the device to be used for small packages that may require secure delivery. Overall, this autonomous vehicle allows for the more efficient use of human abilities and better equips the selected building for the disabled community.

MOTIVATION

This project was originally proposed by David Meek, supervisor of Baskin Engineering Lab Support (BELS). Beyond lab support, BELS is responsible for receiving and delivering incoming packages for the labs at the school of engineering. As the volume of packages increases, it has become more difficult to support the labs. The motivation behind the AMPS project is to develop an autonomous delivery system that could help support the package responsibilities. While delivering packages was the premise of our project, it has become increasingly clear that this system could be used to port other things, such as people. It is our hope that this system could one day be used as an aid to the visually or physically impaired.

OBJECTIVE

The purpose of this project is to build an autonomous vehicle that will deliver a payload to a corresponding location within a building, such as Jack Baskin Engineering. The user will enter a series of package locations into the web interface. Once all desired locations have been entered, the command center PC will plan a route through the necessary checkpoints. The calculated route is then transmitted to the vehicle and activates the start of the delivery process. The vehicle will intelligently navigate the predetermined route using a Cartesian coordinate position tracking system coupled with location checkpoints. Once the vehicle reaches a destination, it will send notification of its arrival to the command center. Once tasks are accomplished at the destination, the vehicle proceeds on route until all payload items have been delivered.

APPROACH

USER INTERFACE AND NAVIGATION

In many applications involving the use of an autonomous vehicle, a line following system or GPS system is used. Unfortunately, a GPS is only feasible when outside where satellite connectivity is available. Since our vehicle will be located within a building, GPS is not a viable option for our project. A line following system requires a network of lines throughout a building that must be maintained. As a team, we decided this would be cost ineffective, cumbersome and would require the cutting of large of red tape associated with building modifications. Instead, we have decided to create a new form of navigation that is both cost effective and required little modification to the building.

Figure 1 - QR matrix barcode

In order to determine the vehicle's location within the building, a series of checkpoints will be associated with the building. The checkpoints will

be in the form of QR matrix barcodes such as those shown in Figure 1. The QR barcodes can be generated at minimal cost as it costs no money to generate through an online website, only the printing portion would factor into monetary cost. It also would require minimal installation efforts to the building, which would include printing and hanging the QR codes on the ceilings of hallways resulting in low labor costs. The QR codes will be located on ceilings to minimize maintenance and possible tampering.

The checkpoints are processed by the autonomous vehicle using an on-board computer. The use of the PC is to allow for the barcode processing through the use of open source software and a basic home webcam. The idea is to have the PC perform the scanning of checkpoints and issue control instructions to the microcontroller which in turn will adjust our motors based upon the instructions associated with that checkpoint for the current route. The microcontroller will have an independent feedback control loop described in the feedback section. The barcode processing would be ran in parallel and only interrupt the control loop when required.

To read the checkpoint codes, an open source GNU LGPL licensed software suite, ZBar, will be used to capture and process camera images on the on-board/client computer. The ZBar software API provides sufficient functionality and accessibility as it is designed to allow access to the internal layers of functionality ranging from low level processing to the high level method calls to invoke image processing. For out purposes we had decided to implement the ZBar software using the high-level access methods. This proved to be incompatible with what we required from it as the processing of barcodes would be running in parallel with the main autonomous process as threading the process caused run-time errors. The recurring error caused unexpected termination threads while processing images. After further investigation, the software bug was a known issue in the ZBar API version being implemented. To resolve this issue instead of implementing the API directly, the given standalone executable running which launched and ran the scanning software was invoked directly using a system call and stayed opened in

parallel listening to scanned codes which piped directly into the autonomous mode control software on the on-board PC.

An additional use of the autonomous mode control software on the on-board PC was to allow for route error checking while in motion. After loading the currently assigned route and activating autonomous mode, the program is expecting to find only the codes on the route in the calculated order. If the vehicle would encounter an invalid checkpoint during navigation, the robot would be commanded to turn around 180 degrees to attempt to find the missed point. If another invalid point is found on route the vehicle stops and engages manual mode two missed point would imply an off-route location indicating that it is lost.

The use of a checkpoint system is twofold. It will primarily serve as the main identifier for destination recognition and localization. The QR codes will identify room/room clusters, hallways, major junctions, floors, etc. And as already described, the codes will also serve as the vehicle's map to the building. The route planning software's task would now be to take the start point of the vehicle, the delivery location(s), and create the best route for the vehicle to take based upon where the checkpoint locations will be. This would be done by applying minimal path algorithms to the graph constructed from the checkpoints themselves. However this will not just work immediately after installation. Each checkpoint must be correctly entered into the database in order for the command center to generate the route that is later interpreted by the on-board PC.

All route planning will happen at the command center. The command center will manage all data associated with the building mapping, such as all checkpoints data correlating with those points. The data will be stored on a database system to allow protected backup as well as flexibility within the various software modules and system overall. Map data is stored in a SQLite database, which is a local, flat-file storage system which uses SQL syntax for data population and queries. The SQL scheme used can easily be adaptable if deployed to a centralized SQL server as the structure of the database will remain the

same. The structure of the database is as follows:

QRCHECKPOINTS *(qr code, bldg. name, x-coord, y-coord, floor number, description, point of contact, deliverable location?, start location?)*

NEIGHBORS *(qr code, north point, south point, west point, east point)*

The database only consists of two tables. The first table contains all information associated with the actual QR checkpoint as filled by the user. The tuples in the database are uniquely identified by the code alone to prevent duplicate codes. The neighbor table relates the adjacent points as related to the given QR code. This table will serve as the primary data provider to route planning.

First, population of the database is necessary and is made through the graphical user interface (GUI) which is in the form of a webpage. The GUI allows for addition and removal of points via an interactive floor plan. The floor plan is a two layer image, the top layer being the clickable portion which is highlighted and in our case is the hallway of the second floor in Baskin Engineering. The user may click anywhere on this highlighted mask. A popup requesting the checkpoint information required to fill in the QRCHECKPOINTS table is displayed. This is how the user may add a point. Upon addition, the new point will associate neighboring locations by scanning the X-coordinate and Y-coordinate ranges respectively until reaching a point or a wall (a wall is indicated by running off of the clickable mask as this is recognized as a transparent background). The new point is then indicated by pin on the map. Clicking the pin will display the associated information. Clicking an existing point will also allow for point removal. If you choose to remove a point the neighbors of the point being removed are updated after the point's removal to ensure data integrity. After having added points, the route planner will now display all deliverable points and starting locations.

The desired destinations for the vehicle will be indicated via the graphical user interface by the user. The background Java process will accept these destination points as well as a starting location which is also indicated by the user. The location codes are used to access the database to obtain all related point

information. Route planning is performed via a multiple iteration algorithm of Dijkstra's shortest path algorithm. In order to generate the shortest route covering all destinations, Dijkstra's algorithm is ran on all possible destinations individually using the initial starting location as the source. Once all distances are calculated the shortest is taken and the path taken from the source to the shortest distance point is printed to the route file. That destination point is then removed from the possible destinations and is set to be the new source point. The algorithm is the repeated on the remaining destinations using the new source. Once all destinations have been visited, a single last destination is implicitly added which causes a final route-back to the original start point. During each iteration, when printing the individual route points, the direction which would direct the vehicle to the next point is also concatenated. An example of the code and instruction may look like this: BE:2:225;STOP or BE:2:I1;RIGHT. The direction instruction is known as we have relation information of where the checkpoints are located based on each current checkpoint code. After calculation is complete, the route is exported to the vehicle for the on-board PC to parse and follow. The route will be manually initiated by the operator at the predetermined start location at which point the on-board system processing takes over as described above for navigation.

Once the vehicle has been initiated and is on route, the UI will double as the vehicle locator. The on-board PC, having Wi-Fi connectivity, establishes a connection to the command center PC which runs the user interface. The route files are received via secure file transfer from command center to the PC. It also sends information via secure file transfer invoked from the autonomous control program. The data sent contains live update statuses to inform the user about the current location and route. It would inform the user about delivery attempts, recently scanned code, missed code, and termination. The status then appears on the live look-in page on the GUI in the form of a banner label. In a situation where the operator needs to find the vehicle, either due to malfunction or other reasons, the user may use its checkpoint status update and/or use the standalone wireless IP camera located on the vehicle itself. The vehicle will be equipped with an IP camera which can be operated through the UI, independent of the

client system. This will allow the user to have eyes on the vehicle at all times without having to depend on the rest of the system module.

SENSORS

As designers, we recognize that an autonomous vehicle traversing a path is nontrivial. How can we guarantee that the autonomous vehicle will traverse a path and avoid colliding into objects? To answer this, we ask: how do humans avoid collisions with objects? One answer is the human eye which provides depth perception. Depth perception is an extremely useful tool in collision avoidance because the information can be easily translated into distance. This is analogous to a sonar sensor. Sonar sensors are essentially the eyes of the vehicle. It sends out a sound wave, and waits for the wave to collide with an object and reflect back towards the source. It calculates the time it takes for the wave to return and provides a corresponding digital value. This digital value can be used to determine distance to an object, essentially giving the vehicle a sense of depth perception.

For our particular application, we have identified two different types of sonar sensors that perform different jobs. One type of sonar will be essentially a wall tracker. We need a sonar to detect the distance between the wall and the autonomous vehicle itself. The other type of sonar is detecting an object in the path of the autonomous vehicle.

The sonar used for wall following should have an effective range of 3 meters to compensate for large hallways in different environments. We also need a way to read close range as well. The saturation region occurs when the sonar is too close to an object to receive the transmitted reflection from the wall. When this happens, the sonar gives a fixed reading. The sonars currently saturate at 20cm. Since not every hallway is made the same, a low saturation point will be useful for narrow hallways. Otherwise, the vehicle may be blind while traveling the narrower hallways.

The other type of sonar we will be using is one with greater distance. These sonars will be mounted in the front and back of the vehicle, ensuring objects in the path of the vehicle will be detected ahead of time. These objects can be stationary objects, or a person who happens to walk right in front of the vehicle. Either way, we want to be notified if there is an object in front or behind the vehicle ahead of time so that we can safely hold the vehicle's position.

The type of sonar sensors we are using have been provided by MaxBotix. MaxBotix specializes in a wide range of sonar sensors including indoor and outdoor sensors. The type of sensors we have chosen is their XL-MaxSonar-EZ/AE line.

Figure 2 - XL-MaxSonar-EZ/AE

Figure 2 is a pictorial representation of the sonar we are using for this design. This device is small, roughly a cubic inch, which will be kept as a low profile on the vehicle. The ranges of this particular line of sonars vary in range from 20 cm to 765 cm.

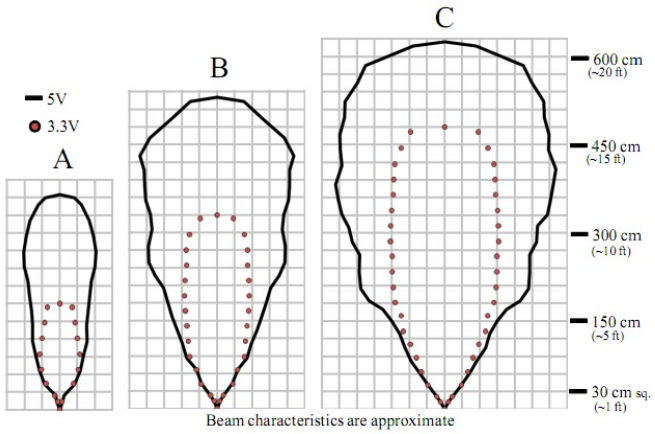


Figure 3 - Beam characteristics for EZ line

In Figure 3, each graph is denoted by a letter: A, B and C. These graphs are determined by placing a dowel at different location. The wave denoted in a solid line represents the sonar being powered with 5V while the dotted circle represents the sonar being powered at a lower voltage of 3.3V. In A, the dowel is place at the closest distance of out the three. The beam pattern is narrower because we detect the

object at an earlier time as opposed to B and C. In B, the dowel is placed further than A, but closer than C. The beam is wider because the sound wave is radiating towards the source is in a concave waveform. Since the dowel is placed further, the concave waveform widens as it traverses further before it detects the dowel. The dowel is placed the furthest in C. The beam detects the dowel roughly at the 20 foot mark. The width of the beam should be noted being extremely wide compared to A and B.

This line of sonars by MaxBotix also varies on the model type chosen. For our application, higher accuracy from the sonar is needed. Based upon the weight of the vehicle, and these sonars serving as its sight, higher accuracy will ensure an object is seen correctly. This will also ensure corrupt data is less likely viewed.

In order to verify the accuracy of our sonar sensors, we tested it by having a team member retrieve real time data by traversing through the hallway mimicking the movement of the vehicle. To certify our predicted sonar choice, the MB1200 and MB1220, will work, extensive testing occurred. The testing was done to ensure how the data looked while stationary, moving, and how different objects affected the beam width. An interesting discovery arose from those tests. The discovery was that organic matter, such as humans and clothing, were not picked up at the angular range the sonars datasheet said it would. The reason being is because when they took the data for the sonar, they tested using a wooden dowel, which reflects the energy back better than human clothing, such as jeans, and human skin. After this test, we determined that the beam width will not be as wide as we originally anticipated. However, this only changed how we would need to place the sonars on the vehicle.

Due to this discovery, we decided to place 4 sonars in the front of the vehicle, with 2 of them angled slightly outward (roughly 30˚). This also forced our choice on which sonars to use. Since we want to anticipate any object in front of the vehicle, the front facing forward sonars will be narrower beamed sonars, the MB1220 line. The angled front sonars will be the MB1200s, which allows for a wider viewing angle and will serve as the vehicle's peripheral vision. All the other sonars, i.e. the side and back sonars,

will also be the MB1220 line to allow for longer distance measurements. A major concern we have with placing the MB1200 behind the other sonars is the increased vulnerability to interference. This interference will be caused by the MB1200, or the MB1220, since they use the same frequency. Essentially, when an MB1200 sends out a signal, while a MB1220 close to the MB1200 is also sending a signal, the MB1220, or MB1220 will receive interference from the other. This will cause the sonar to see nothing, which will cause problems when trying to see if an object is in the way.

To handle the interference, we have tested to see the significance of the interference. The significance was too large to ignore and caused us to see nothing when an object was in the way. To adjust for this interference, we have moved the MB1200 to be on the corner of the bump guard. This eliminated our significant interference, and gave us good data to create a working collision detection system and feedback control algorithm. Figure 4 demonstrates a rough visualization of the "sight" of the vehicle.

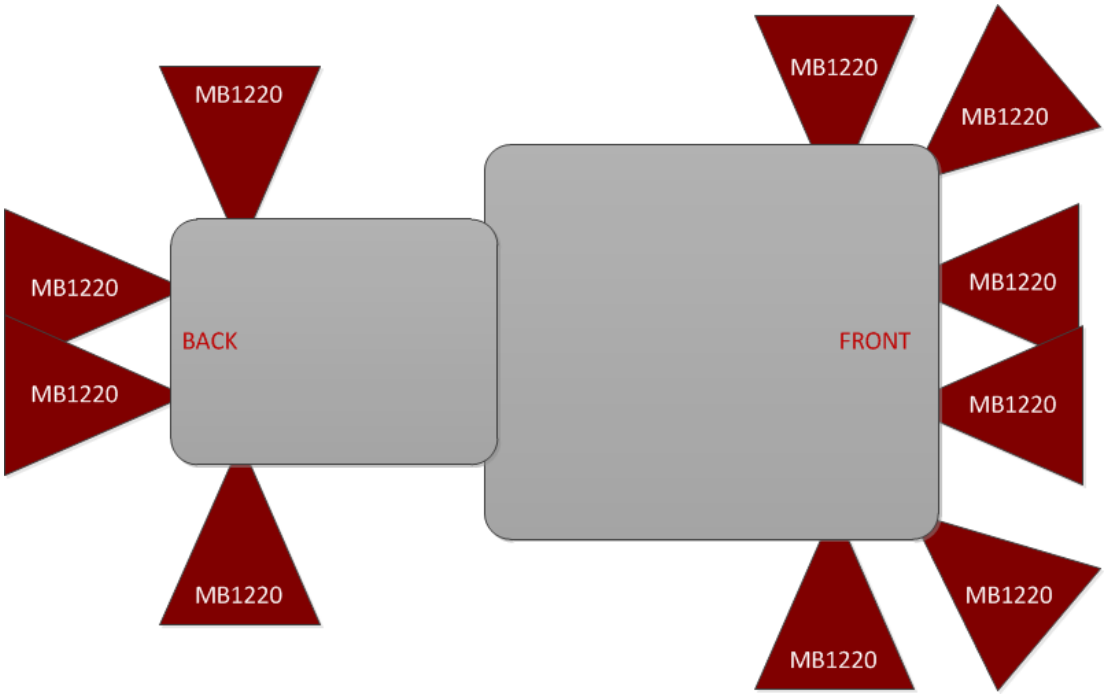


Figure 4 – Visual layout of sonar

Another point to be made with the sonar routing is the type of cables we used. In order to make this more versatile, we pinned out all available pins on the sonar board. Seven total connections will be needed when doing this. However, instead of making a special connection, or soldering 7 wires together and running them back to the microcontroller, we created a PCB board that will converter our wires to a standard RJ-45 connection. This also involved creating a second break-out board on each micro to convert the pin out to a RJ-45 connector. Using standard Ethernet wire we can run each individual connection to the main daughterboard, where the microcontroller will be attached. Using this type of wiring, we are allowing for more versatility, and much easier to replace wires, should they become damaged at any point.

As shown in Figure 4, we have ten different sonars places on the perimeter of the bot. Interference becomes a problem since each of the sonars can only produce a 42kHz sound wave. The sonars will not be able to differentiate between the sound wave one sonar produces, and its neighboring sonars. Therefore we need to implement a way to control the pulses so that we can guarantee that the sonar who is pulsing out the 42kHz wave is the only one receiving it. We do this by daisy chaining the sonars.

Daisy chaining ties multiple sensors together and only one device is active at any given time. The sonars obtained from Maxbotix gives us that ability to control the sonars. Normally, we would just leave Pin 4, the RX pin pulled high, which continuously ranges and collects data. In the daisy chain configuration, we want to control the RX pin to let the sonar know when we want to range. In Figure 4, we will use the back sonars as an example. When the daisy chain starts, the microcontroller sends out a 20µs pulse to the RX pin of the first sonar. It takes the sonars 99ms from the pulse entering its RX pin to its completion to obtain data. Once it is completed, the TX on Pin 4 of the first sonar sends a 20µs pulse to the second sonar RX pin. By doing this, the first sonar is now off because its RX pin is low, and the second sonar is on. Another 99ms goes by and the second sonar is done ranging. The TX pin of the second sonar

sends a pulse out that goes into the first sonars RX pin. This completes the daisy chain. The two sonars will continuously range at mutually exclusive times until the power is turned off.

Sight is not the only way to navigate. If we decided to only use a sonar system, then we will be forced to rely on solely upon our software to tell us if the vehicle is moving or not. This is problematic as the software will have no data verifying movement on the vehicle. Our solution to this is to incorporate encoders that will monitor the rotation rate of each wheel.

The encoders are made of thin gauge aluminum that is then mounted to the shaft of each wheel. This ensures that the encoder wheel is locked to the rotation of the actual tire. As the encoder moves, an optical encoder is used to monitor the small slits that are seen in figure 5. As the encoder passes over the each slit, a small pulse is generated and captured by the microcontroller. Given that each tick on the encoder represents a finite degree of wheel rotation, we can use the pulse width captured by the microcontroller to determine the precise rotation rate of the wheel through the following equation.

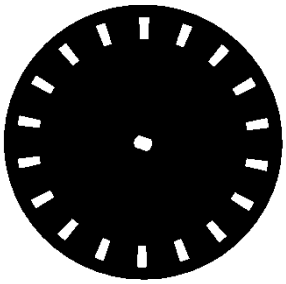


Figure 5 – Encoder Wheel

$$\square\square\square\square * 360 * 1\Delta\square = \square\square\square\square\square$$

Sigma is your degrees for encoder tick, for out design this is 5^{o}, and the time delta is measured by the microcontroller. The speed measurement inherits the units used in your circumference. This system allows us to monitor wheel speed many times in a single rotation which is necessary because will be moving very slowly.

While we are designing the system to autonomously navigate without hitting anything, it would be irresponsible for us to assume this would never happen, this is why we need both collision avoidance and detection. If for some reason the sonar fails to pick up an object ahead of the vehicle's path, we

would be able to detect it through bump sensors. The bump sensors are designed to be our final and most reliable line of defense in the event of a collision.

The sensor of choice was a force sensing resistor (FSR). This particular sensor has a variable resistance that is dependent upon the applied force it receives. As the force applied to it increase, the effective resistance decreases. The slightest collision cannot be detected through monitoring the effective resistance of each sensor on the bump assembling. In order to generate a varying voltage, instead of resistance, configure the FSR into a voltage divider with another finite resistance. As we apply force, the resistance changes, which directly affect the voltage out of the divider circuit by the following equation.

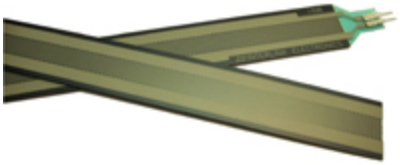


Figure 6 – Force Sensitive Resistor

□□□□=□□□∗□□ + □□□□

Where R is the series resistance you placed in the voltage divider. The varying voltage is fed through a comparator circuit to decrease the transition time and time it takes for our system to stop. Each FSR circuit is run in parallel, meaning any collision along the perimeter of the vehicle will be detected.

Using the FSRs proved to be much more difficult than we originally imagined. Unfortunately, the FSR sensors are highly sensitive at their connection joints. To connect them, we soldered leads to the small silver pins at the end of the sensors (see Figure 6). Our plan was to cover this portion of the sensor with shrink wrap to keep the connection strong and protected. However, the plastic portion next to the leads protects other sensitive parts of the sensor, which gets damaged when using heat to shrink wrap the leads. Unfortunately, we were forced to use electrical tape to keep the connections protected.

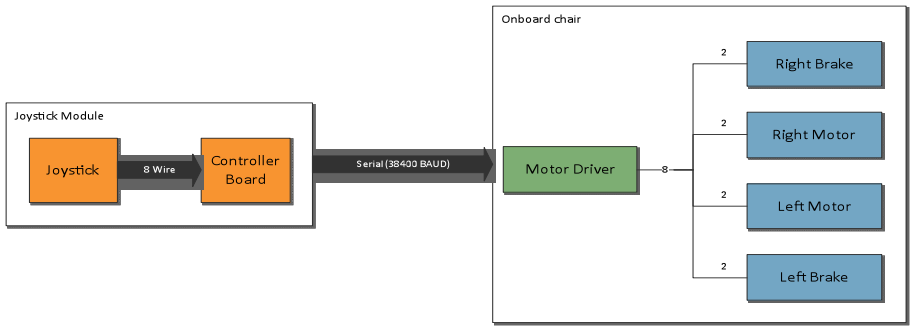
Another issue we ran into with these sensors was applying them on the bump guard. The FSR sensors require the connection leads to stay straight, with minimal bend on them. When placing them on

the bump guard, we need to slightly bend these to run them to the circuit that will control the relay to the motors. By placing the sensors like this, it will decrease the longevity of the sensor. To compare to when the leads are left straight, the longevity is 10,000 bumps, while bent lasted only a few weeks. For this reason, we adjusted the connections to make sure the leads are not bent and allow for maximum longevity.

When applying the FSRs to the bump guard, we need to take another consideration into account. The FSRs have a sticky side to apply the sensor to a system like ours. When applying the sensor, there cannot be any air pockets left underneath the adhesive. If air pockets are found, there will be extra stress on the sensor and will cause a different reading on the resistance when reading no pressure. Careful placement took place to ensure this did not happen. To further ensure no air pockets occurred, and to make sure they were still working, we measured the resistance of each sensor after placement.

MOTOR CONTROL

- Onboard Driver
- Onboard Motors
- Controller Hardware



File: Motor Block Diagram.vsd
Project: Autonomous Mobile Porter System
Date: 03-14-2011

Figure 7 - Motor control

The wheel chair comes equipped with two high torque DC motors. Each motor is equipped with a solenoid breaking system. Each motor/break system is driven independently from the VR2 driver module. The system follows the block diagram in Figure 7. As you manipulate with Joystick in the control module, the movements are translated to a particular bit stream which is send to the motor driver module. The driver module interprets the bit stream and produces pulse width modulated signal that is sent to each motor. For our purposes, the chair must be controlled by a microcontroller. Given the current system, there are 3 points where this can be accomplished. Option 1; manipulate the control signals being sent from the joy stick to the controller, this in turn will generate the proper bit stream to move the wheel chair. Option 2, bypass the joystick controller and generate the proper bit stream with a microcontroller which will be interpreted by the motor driver and drive the wheels. Option 3; bypass both the motor controller and driver by driving the motors directly from the batteries using an H-Bridge.

Given the lack of documentation on the controlling system, options 1 and 2 would require that we determine the proper signals experimentally that our microcontroller must mimic. Give the extreme time requirement of our project, this is not advised. This being the case, option 3 offers the simplest and fastest solution.

In the interest of time, we decided to design the board with off the shelf H-bridges. In order to purchase the correct module, we had to gain a working understanding of how the motors functioned and what sort of operating characteristics they had. Not knowing the complexity of the of the native driver system, it was nearly impossible to monitor the current for each motor directly. Since the motors produce the majority of the current draw of the entire system, we monitored the drawn current of the existing system by placing an amp meter in series between the two batteries. From the total operating current, we could extrapolate the operating current for each motor.

It became increasingly clear that these motors were capable of drawing a significant amount of current if you were anywhere close to driving them at 100 percent, well over 60 amps per motor, at stall.

Given that we decided to purchase our H-bridges, he had to make the decision that we would not find an H-bride module capable of switching that much current. This was not as large of an issue as it sounds. Given the purpose of our system, we would be moving at such a speed that would require no more than 15 amps per motor. While we maintain that this is still a significant amount of current, we were able to locate modules capable of switching 15 or more amps.

After running a number of bench tests in the lab, it was simple to see that these motors could be very dangerous if not treated with respect. Seeing the potential threats the motors could offer, we designed the board with the thought in mind that we had to always maintain a mechanical disconnect when they are not in use. The driver board is designed with a large DC relay. The relay offers has two desired properties: one, there is physical disconnect between the batteries and motors whenever the board is not activated, two, the relay is capable of switching extremely high currents while offering isolation from the actual voltage controlling it, in our case the bump sensors. The board is designed in such a way that it fails off. In other words when the vehicle is first powered, the board must be activated before the motors can be moved. In addition, whenever power is lost to the system, the board will disengage power from the motors. It is important to realize that this safety system is completely independent from the microcontroller and is immune to software errors.

It became clear during the progress of this project that ground plays a far more vital role than originally anticipated. In the process of integrating the microcontroller with the motor board, we ran into a scenario where the motors would drive independently in different directions but were incapable of driving be driven in the same direction. Further investigation with an oscilloscope showed that we had some signal integrity issues. The PWM signal would appear high distorted when driving the motors in the same direction, but was fully intact when they were driven in opposite directions. We very quickly determined that this was fixed by sharing a direct ground path between the microcontroller and driver board, though this was not a viable solution. By sharing a direct ground path, the high transiency of the

motors being switched off during a collision had the potential to crash the microcontroller. We found quickly that a solution to both issues was to ground the metal box containing all the components. It is a bit unclear why the grounding of the box fixed the issue.

MICROCONTROLLER

For this project, we decided to use a microcontroller in conjunction with a fully function PC. This solution gives us the flexibility of general purpose IO, while maintaining high level functionality of the PC such as wireless internet connectivity and web camera interface. The micro of choice was the Atmel ATXmega128A3 microcontroller. The controller came equipped with 16 analog to digital converters and 16 GPIO's that could be configured for input or output capture. This was more than enough to suit our needs for the original design.

Since the microcontroller is used to gather the data from the sonar sensors, we needed to understand how the data was gathered from the sonar sensors. Through code, we establish settings to analyze the data correctly. We establish the analog to digital conversion, along with the resolution we would like to receive. In our system, a 12-bit resolution is used. In order to scale to centimeters, Maxbotix recommended that the sonars be read in as a 10-bit value. Therefore, any values that were read from the ADC was bit shifted to the right to lose the lower two LSB, essentially converting it to a 10-bit value. Through testing, we obtained a linear offset from our readings that were inherent within our microcontroller. This was simply corrected through software.

The motors are also controlled with the microcontroller. Since each motor is a DC motor, we need to be able to control how much DC they take. Using the microcontroller, we establish a varying PWM wave to control the speed of the motors. Each motor is connected by an individual H-bridge, which allows forward and reverse rotation. The microcontroller is used to adjust the PWM to each motor through our control functions. Within this part, we establish a maximum PWM that can be used on each

motor. Before we can adjust PWMs to the motors, a frequency needs to be set for us to adjust. The motors can run off of various frequencies; however, we chose a value that will not interfere with our sonar sensors, and was the previous frequency used by the VR2 controller on the wheelchair, which is a frequency of 20 kHz.

For further control, we created functions within our code to control our feedback loop, ramping up, ramping down, and turning. During manual control, between every command given, a stop function is initiated. This ensures the motors will not reverse direction instantaneously, which will cause serious damage to the motors. During autonomous mode, the motors are controlled through the feedback loop and state machine, which account for any instantaneous changes in direction as well.

Safety precautions were also made to the motors during both modes. These motors have the potential to draw as much current as they feel necessary. The reason for this high current draw is the fact that they are high torque motors capable of carrying heavy loads. The motors adjust the current based upon the speed given by the controller and weight on the wheelchair. This was apparent when tested the stall currents of the motors. We control this draw of current with the PWM of the motors. By limiting the maximum PWM the motor can use, the maximum current they can draw is also limited.

Since the microcontroller controls the motor speed and direction, a state machine was developed to control reactions and feedback loop of the system. The state diagram is described in Figure 8.

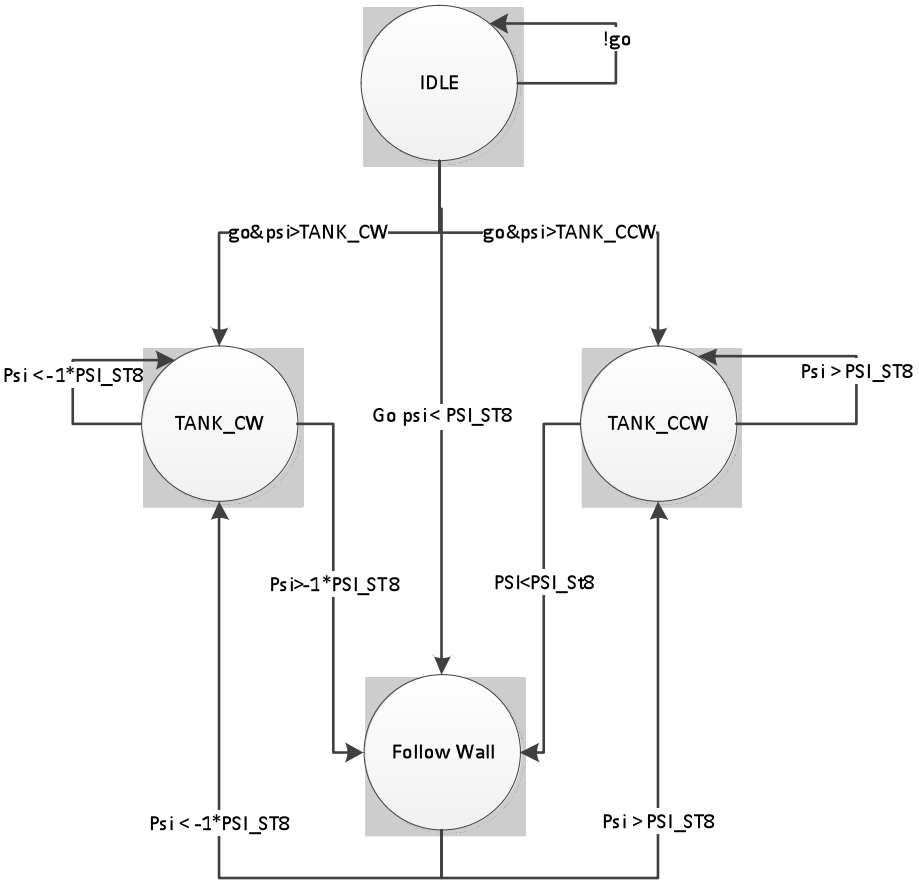


Figure 8 – Control State Machine

1. State Machine

    a. IDLE: The system is a motionless state and awaits a go signal to be assessed from the onboard PC.

    b. TANK_CCW: This system is rotating in a counter clock wise fashion in order to find a parallel heading with the closest wall.

    c. TANK_CW: Rotates in clock wise fashion, follows behavior from TANK_CCW.

    d. FOLLOW WALL: The system will be actively tracking a wall and making its way down the hallway. This is the state that implements the feedback controller.

FEEDBACK CONTROL

As stated before in the Navigation section, the data gathered by the sensors will control our motors. This is done using a PD controller, implemented within code. However, before we can discuss the code involved, we need an idea of what calculations need to be done.

For our purposes, we want to move parallel to a wall using the data gathered by the sonar sensors. After a few meetings with Professor Gabriel Elkaim, we had a much better understanding of how to approach this problem. Our first step was to determine the type of sonar we are using. For example, some sonar sensors will measure what is directly in front of them, regardless of angle. These types of sonar sensors act more like infrared sensors in how they read the object. Unlike an infrared sensor, our sonar sensors will read what is directly in front of them, as long as the object is within the sensors conical range.

Before we can move onto the actual calculations, we need to understand the theory of what we are trying to do. For all applications, we are contained in what is called "world coordinates," which consist of longitude, latitude, and the compass. World coordinates ware useful when using a Gyroscope, or a compass, but we do not plan on using something along those lines. In our system, we can transform these coordinates into our segment coordinates. The segment coordinates put a perspective heading on our vehicle based upon the data gathered; therefore, there are no compasses, only orientation with respect to a wall.

With this part determined, we can now move onto how we can align ourselves with a wall. This can be calculated with using 2 of our side sonar sensors. Based upon their data, we can determine a heading, which we will want to drive to zero. Figure 9 is a visual representation of how we plan on obtaining our heading.

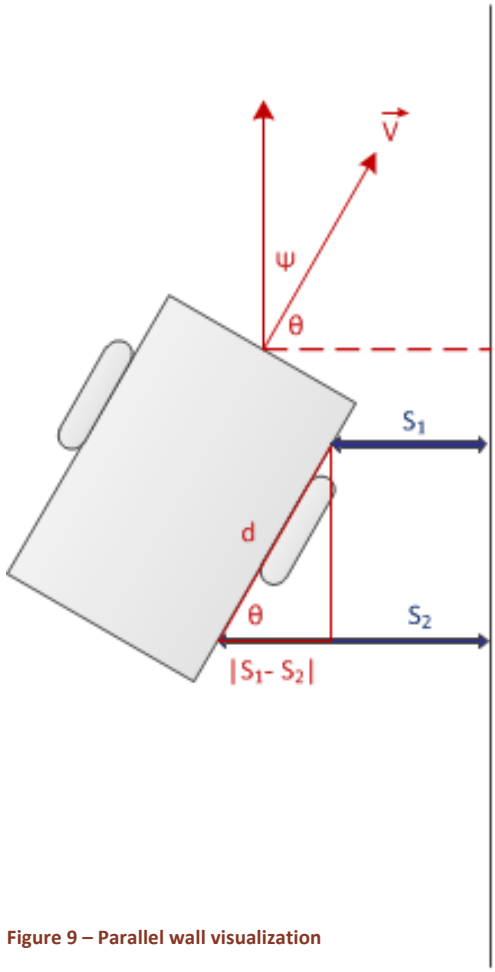


Figure 9 – Parallel wall visualization

Based upon Figure 9, we can calculate our heading (Ψ), or error, if we know the distance from the vehicle to the wall. Our sonar sensors measure this for us, so we only need to apply our data to equations in order to determine the heading. For this diagram, our heading is found using these equations:

$\theta = 90° - \theta$

In order for our system to stabilize, we need to drive this value to zero. This implies that θ needs be pushed to 90°.

$\cos\theta = \theta 1 - \theta 2$

$\theta = \cos -1\theta 1 - \theta 2$

Our equation for Ψ now changes to:

$$\psi = \tan^{-1}\frac{y_1}{y_2}$$

Once the heading is calculated, we can adjust it by changing the percentage of a PWM wave on a single motor. With this information, we can now create a feedback loop that will adjust the motors based upon the heading. Figure 10 describes the feedback loop as a block diagram.

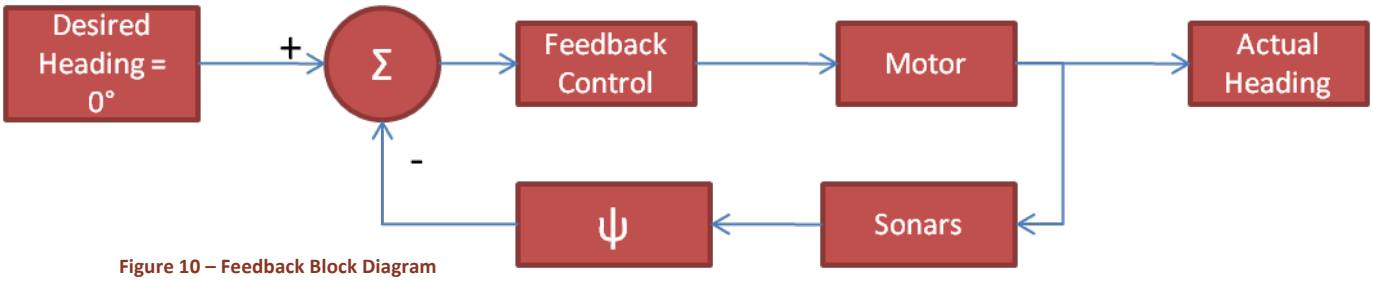


Figure 10 – Feedback Block Diagram

Our error calculation that the PD controller will adjust for the system is now:

$$Speed = error + error + error$$

Where *Speed* is the current measured value of the PWM, the *error* is the measured difference from the sonars oriented to the wall, K_p is a constant kept under 1 and weighs the error, and K_d is a constant also kept under 1 and weighs the error calculated. The reason some of the errors are waited is due to how much we want the system to respond to certain errors, such as a small change in the sonar data will not force the robot to adjust quickly. Our feedback equation is for a PD controller, which controls our system's stability when centering down a hallway. However, before we start adjusting the motors and sending it down a hallway, we need to simulate the adjustments made through our control equations. Using Matlab, we code our PD equation and adjust values to verify our stability while traveling down a hallway. Our Matlab stability is described in Figure 10, where the values of K_d and K_P are .01 and .7, respectively.

If we adjust the values of K_d and K_P, we can adjust the reaction of our PD controller. The larger K_d gets, the more overshoot and unstable the system gets. Also, if we adjust K_P to be larger, the PD

controller reacts on lower errors, creating a more sensitive system, and possibly more unstable. Under simulations, we can establish the correct values for this system.

The actual implementation showed to be much more difficult than we originally anticipated. We developed a single PD controller that was designed to drive the given error to zero. What we found to be most challenging was that we had multiple errors that needed to be handled. We needed to maintain a relative heading of zero with respect to either wall of the hallway. In addition, we needed to maintain a safe operating distance from either wall. Lastly, we needed a method of avoiding objects within the hallway. There were two method of correcting these errors. The first option is to have a separate PD equation for each error. The second option being to have a single PD equation that corrects multiple errors, we chose the second option. Each error is generated and is summed together into one weighted error. The most important error we must account for is the heading error, this means that it was given a heavier weight in the summation. Once this error is calculated we can insert it into the aforementioned PD equation. The PD equation was given control over the PWM's to the motor, thus adjusting the wheel speeds until the desired heading and distance from the wall is reached. The last thing we did was limit the maximum and minimum PWM values that the PD equation could generate. This resulted in the robot travelling in much smoother fashion.

POWER

Based upon our design structure of an electric wheelchair, power will be essential. To elaborate on how power will play a larger role in our design, we need to clarify the constraints involved with our project. These constraints are: efficiency, reliability, and heat emission. Usually, noise would also be an area of concern, especially with sonar sensors; however, using the XL-MaxSonar-EZ/AE line, noise filtering is already established.

Our first constraint is efficiency, which is a key aspect to the design because it will affect our lifespan. The base model of our design, the Jazzy 1420, can last about twenty miles before a charge is needed. To be realistic, we feel the vehicle, after the entire build is finished, should last at least 8 miles before a charge is needed.

According to the user manual on the Jazzy 1420, at full charge, the wheelchair should last for up to 20 miles before the next charge is needed. The manual also provides us with a typical max speed of 4.5 mph. With this information, we can calculate the amount of time the Jazzy 1420 should last if traveling at full speed.

$$20 \text{ miles} * \frac{1 \text{ hour}}{4.5 \text{ miles}} = 4.4 \text{ hours}$$

. We can now use this as our maximum efficiency. As long as we are close to this value, we should last long enough for our vehicle to complete all the deliveries in a day.

With this in mind, we need to determine how much current draw all of the components will take. The batteries on the Jazzy 1420 are two 12 Volt (V), 73 Amp hours (Ahrs). These are connected in series, supplying 24 V, with 73 Ahrs. Knowing this information, we can now look at the restrictions each component of our system will need. Our power budget is as follows:

Component	*Required Voltage (V)*	*Required Amps (A)*
XL-MaxSonar-EZ/AE	3.3 – 5	.05 - .1
XMega1283	1.6 – 3.3	.1
Encoder system	3.3	.1
Bump sensor system	5	.05
IP camera	5.5	2
Safety beacon	12	.7
Zotac ZBOX Plus	19	3.42
Motor system	24	10

Table 1 – Power Budget

Totaling all the current required for our system, including the use of 10 sonars and 3 IR sensors (from encoder system), we will have a max current draw of approximately 17Amps (A). This was calculated with the sonars being supplied 3.3 V, their minimum requirement for accuracy. This was decided due to the constraints on the XMega1283 microcontroller, which only allows up to 3.3 V on the input of its input capture pins.

We can calculate our efficiency by dividing the amp hours of the batteries by our total hours to gives us how many hours before the batteries cannot supply anymore current. The resulting efficiency of our system is now calculated to be 4.3 hours per charge, which is fairly close to our max efficiency. From this, it would appear that all of our additions do not drastically affect the system's efficiency.

Realistically, we will not be traveling at max speed for safety concerns. We can recalculate our efficiency using a smaller traveling speed. For our purposes, we should not be traveling any faster than about 2 mph, max. This will decrease the amount of current drawn by our motor system, which will increase our efficiency. However, now our additions will affect the system efficiency and will place a limit on our maximum efficiency.

Our second constraint is reliability. As viewed within our power budget, not everything we are installing requires 24 V. To compensate for this, we will be using linear regulators along with a switching converter to create 3.3, 5, 12, and 19 V power rails. These will connect directly to the battery supply and down convert the 24 V to the select rail. Careful consideration needs to be taken when deciding upon each rail system's regulator. For each system, heat emission, our third constraint, will need to be considered because we do not want our components getting too hot.

Our first rail of interest is going to be our 5 V rail. After researching multiple different options on this particular item, a DC-DC converter was found that will convert a voltage supply between 19 V to 36 V into 5 V. The other important thing to note is how much current is needed on the 5 V rail. Adding up the current needed on the 5 V rail, we have a total of over 2 A. Due to the high current need, a LM7805 linear

regulator, a typical regulator used for this application, cannot be used. The LM7805 is not able to supply the current required, and to save money, a DC-DC converter is the best option. The DC-DC converter used for our purposes is the SD-25B-5 by Mean Well, Inc. that supplies an adjustable output between 4.5 to 5.5 V and up to 5 A. This suits our needs, and allows for additional 5 V items to be placed, should we need room. As far as heat is concerned for this system, it is rated to operate at 25 Watts (W), which will keep our temperature down on this item.

The next rail is the 3.3 V rail. For this rail, we do not need to purchase a DC-DC converter due to the low current draw. Instead, this rail can be made using an LM1117-3.3, a linear regulator, to converter down to 3.3 V and up to 1 A of current. The LM1117-3.3 requires a 1.6 – 10 V input, in which case we will attach this to the 5 V rail. For heat concerns, we need to calculate our expected heat emission based on these equations:

$$P_{dis} = (T_J - T_A) * \Theta_{ja}$$

$$\Theta_{ja} \leq T_J + P_{dis} \text{ (°C)}$$

Where P_{dis} is equal to the power dissipated on the regulator, T_J is the maximum allowed temperature for the regulator to operate in (150°C), T_A is equal to the ambient temperature (25°C), and Θ_{ja} is equal to the heat emitted from the junction to ambient. All values except for the power dissipated are given from datasheet. For the 3.3 V rail, our calculations result in P_{dis} equal to 1.53 W. Using this value, we calculate the needed Θ_{ja} to be less than or equal to 81.7°C/W. The LM1117-3.3 without a heat sink has a Θ_{ja} to be 79°C/W. This means that we do not need a heat sink, however, this component will still become very hot to the point you cannot physically touch it without getting burned. A typical heat sink for this will change the Θ_{ja} to be around 20°C/W, which now falls to a reasonable temperature when dissipating 1.53 W. This can be calculated by:

□□□∗ □□□□+ □□=□□□□□□□□□□ □□□□□□□□□□□□□

We need to take into account the starting temperature, which is why the ambient temperature is added to our product. For this rail, our operating temperature should be 55.6℃, with a heat sink.

The 12 V rail will only be used to run the safety light. The current requirement on the safety light is .7 A. An LM7812C, a linear regulator, can be used to step down from 24 V. However, now our heat will play a larger factor on the circuit.

Using the same formulas, we find that T_J and T_A are 150℃ and 25℃, respectively. Our dissipated power is now going to change due to the 12 V change in voltage. There is an important characteristic of the batteries that we now need to consider. They are not a perfect 12 V each all the time. Due to their physical makeup, their voltage will change to be as high as 26 V and as a low as 20 V. This is a characteristic of the battery, and will not change our system too much due to our regulation; however, it will adjust our temperature drop over this rail.

Instead of the voltage difference being 12 V, we need to consider the impurities of the batteries, which will jump our drop to as high as 14 V. Following our equations, we reveal that our P_{dis} is 9.8 W. As a comparison, the 3.3 V rail will only reach the temperature of 101.5℃, without a heat sink. The 12 V rail, without a heat sink, will reach a temperature of 515℃ due to the higher power dissipation over the chip. Soldering Irons, usually work at this temperature, so you can imagine having a permanent soldering iron on, and probably melting everything around it. This is also out of our range of operation, which is 150℃ for maximum temperature. With the typical heat sink of 20℃/W, the heat will still be out of our operating range, resting at 221℃. In order for this chip to suit our needs, we will need to heat sink properly.

Following our same formula we used on the 3.3 V rail, our Θ_{ja} needs to be less than 12.7℃/W. Fortunately, heat sinks for the package type we are using (TO-220) are not hard to find. To choose the correct heat sink, we need to look at the Θ_{ja} given by the heat sink, which needs to be less than the 12.7℃/W calculated. For our circuit, we found a heat sink that has a Θ_{ja} equal to 2.6℃/W, which is way under our calculated value. This is actually what we want to do because if we choose a heat sink closer to our value, the chip will operate at the maximum temperature allowed (150℃). Typically, this is still too hot to operate in, so by choosing a heat sink that allows for the operating temperature to be under this value, we can have a better chance of not melting our parts. With this heat sink, we should be operating in a temperature of 79.88℃. This was calculated by multiplying our Θ_{ja} of the heat sink, plus the internal heat of the chip (3℃/W), by our dissipated power.

Our final rail is the 19 V supply for the Zotac ZBOX Plus. The Zotac ZBOX Plus needs a power supply of 19 V, ± 10%. With the 10% variance, we can supply the Zotac with 18.05 V to 19.95 V, and it will operate like normal. However, to make sure it operates like it says it does, we will supply 19 V. Unfortunately, there is no dc-dc converter in production that creates the needed 19 V supply. Instead, using National Semiconductor's workbench for building a power supply, we can evaluate different designs and simulate the design to verify it works. The simulation is helpful because it shows what the system should look like under the various values of the input voltage. This is also helpful because of the high current draw the Zotac uses, 3.42 A.

The design chosen on off the workbench involves the use of the LM3150, and two MOSFETS, to increase the current output. The LM3150 is buck converter, and has an adjustable output voltage based upon the values chosen for the feedback input. For circuit layout, please view Appendix A – Power Rails.

Based upon the schematic in Appendix A – Power Rails (19 V rail), we can calculate the needed output voltage using this equation:

□□□□= □□□□□□□1+ □□□2□□□1

Where V_{out} is our desired voltage output, V_{fb} is our feedback voltage connected through a voltage divider, R_{fb1} is our feedback resistor to ground, and R_{fb2} is our feedback resistor connected to the regulated output of the chip. According to the datasheet on the LM 3150, the V_{fb} will take .6 V to operate correctly. The workbench also suggested values for these resistors, but we should still check using this formula. According to the workbench, R_{fb1} is taken to be 10 kΩ, and using the equation above, we can calculate the needed R_{fb2}. Following the equation above, but instead solving for R_{fb2}, we get the value to be 308 kΩ, which matches what the workbench gave us. The next step is to view what happens to output voltage if we change R_{fb2}. Running through the calculations, our V_{out} changes based upon the voltage divider ratio. This holds because the V_{fb} wants to see .6 V, so it will adjust V_{out} through the switch on the LM3150 until it sees the desired .6 V. It does this by internally checking the V_{fb} through a comparator set to stay between .36 V and .72 V, for overload protection. If over or under this range, it acts like a short circuit and lowers V_{out} until it is no longer shorted. Once it passes through the short circuit analysis, it is pulled to .6 V on another logic component within the chip that regulates V_{out}.

Fortunately, the workbench also allows for us to order the components with a PCB designed for our specified application. The next step is to order the parts and test to see that everything is working as it says it is. Heat sinking for this layout will also be key, but fortunately, the workbench takes that into consideration as well when creating the PCB.

STRUCTURE

Another component to the vehicle is to create a structural design that will safely and securely carry a payload. This would involve determining a max weight requirement for the payload, implementing a locking mechanism to secure the payload, physical characteristics to ensure safe movements, placement of components, and cable routing.

Fortunately, the wheelchair frame gives us a max load to be placed on the wheelchair for passenger transportation. This helps us determine a max payload weight. The max weight of the wheelchair is 500 lbs; however, depending on the structure we build around the framing, this max weight will change. Anticipating a need for structural integrity, we built the storage unit out of various gauges of steel. Adding steel to the vehicle increased the weight on the frame, which decreases our payload weight. The decreased weight for the payload is still not significant enough to slow the vehicle for delivery means, or exceed the original max weight requirements.

Another consideration is how to adjust for weight shifting of the payload. To counteract the shift, we analyzed how storage containers are designed to counter the shift. Leading us to the conclusion that a file cabinet like system is best, we built a steel storage unit to be placed on top of the frame. Measuring 3’ 2” tall, with a 3’ x 1’ 8” base, we installed 2 pullout drawers for easier loading. Another constraint was made when installing the drawers. Each drawer now has a maximum weight that can be applied. This was determined by the specifications on the slides purchased for the drawer system, which is 100 lbs. This still allows us to carry up to 200 lbs within the storage container.

One important part of this portion of the structure is that it needs to be detachable, a key feature should this platform be used for other means, or an electrical problem occurs. The detachment of the storage unit requires taking out our six lock bolts placed on the bottom of the storage unit, going into the frame of the vehicle. To ensure that this doesn’t happen while in transit, lock nuts were used to keep the bolts from getting loose. The prototype in Figure 9 is a visualization of what the storage container looks like fully assembled.

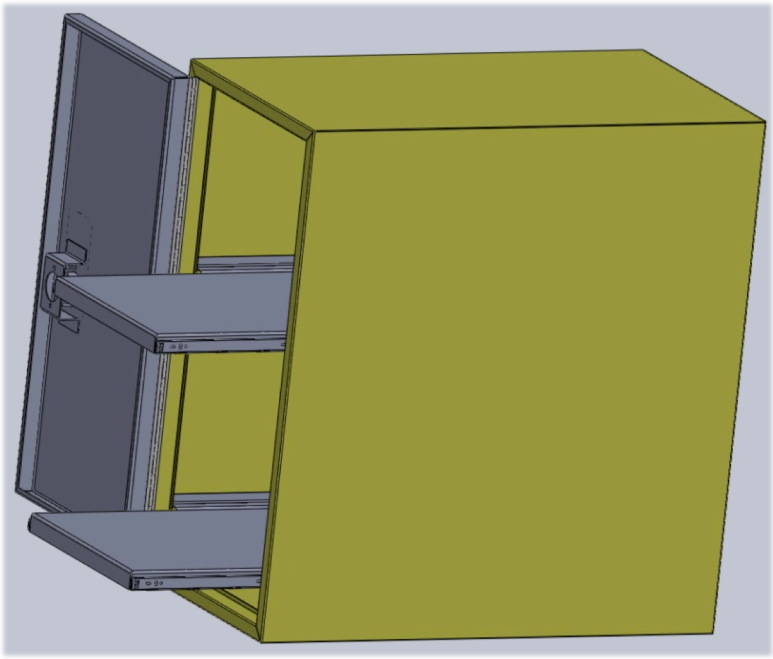


Figure 9 – Storage Container

Once the weight concerns were addressed, we approached the protection of the wheels and components. This involved developing a custom steel bump guard that would surround the vehicle frame. The steel bump guard does not only protect the vehicle, but also protects anything it may collide with. As we have stated before, in an ideal situation, the bump system will never be triggered. However, in case it does, we need a frame to wrap it around so nothing else on the vehicle is damaged.

To ensure the security of the payload, an Omnilock OM500 has been installed. This system uses a key numbered entry programmed into the lock system. Once the correct entry is made, the door will unlock long enough for the door to be opened and the payload extracted. Fortunately, Jack Baskin School of Engineering uses this same system, and was able to donate an older, working model that had been removed for a newer model. This saved us a considerable amount of money as these items usually run around $1,000 per unit.

The next task was programming the access codes into the unit. To do this, access to a specific printer was needed, which facilities at the Jack Baskin School of Engineering made available. Using the printer, we entered the master code and entered menu mode. Through the menu, all functionality can be used. Within the menu is where we were able to program 3 pass codes, one for our entry, one for the receiver, and one for Baskin Engineering Laboratory Support (BELS) to enter the storage container. By placing a specific code for each entry, the overseers (BELS) can use the printer to view who has accessed the unit. This will be a useful tool should a payload become misplaced after being sent out for delivery within our vehicle.

In addition to the security lock, a safety light and wireless IP camera are also used. The safety light is used to ensure visual notification is made with any surrounding traffic. The IP camera, which can be used to view where the vehicle is, can also be set to record while out on delivery. This will see all items that are in front of the vehicle and can swivel up to 270˚ to view some objects behind the vehicle. Due to the IP cameras recording ability, a sign will be placed on the storage unit to notify individuals that they are on camera.

Placement of the components is major concern. We would like to have a designated spot to place the components of the design, similar to what we have built for the storage container. However, since we would like to make this system work without the use of the storage container, placement of the components needs to be localized to the frame of the wheelchair. Analyzing this portion, we took measurements of the needed space, with what components need to go where. Our list of components are: on-board PC, encoder circuit, bump circuit, motor board, Ethernet daughter board, 5V power supply, 3.3V power supply, 12V power supply, and 19V power supply. For better isolation from noise and heat, all the power supplies will be placed into their own container, located beneath the storage unit, toward the middle of the frame. All the power connections will then run to their needed connection points. The

other components are then placed upon the vehicle in the foot-rest. A custom steel container was made to house these components.

In order to ensure that wires are not tangled, interfering, or causing unforeseen errors, careful layout of our cable routes was used. To analyze this better, we isolated certain areas of the vehicle. On the front wheels, we need to keep wires out of the way of the motors and wheel movement. To fix this, we placed cable ties that screw into the steel bump guard around the wheelchair base. The back wheels where done the same way, routing the cables on the steel bump guard. Instead of routing everything around the batteries, we measured the clearance the vehicle had. Fortunately, the clearance was high enough that we could place plastic covers underneath the battery tray on the vehicle. This is useful since we would like to place more of the components in the front of the vehicle. Figure 10 visually depicts the final prototype of vehicle.

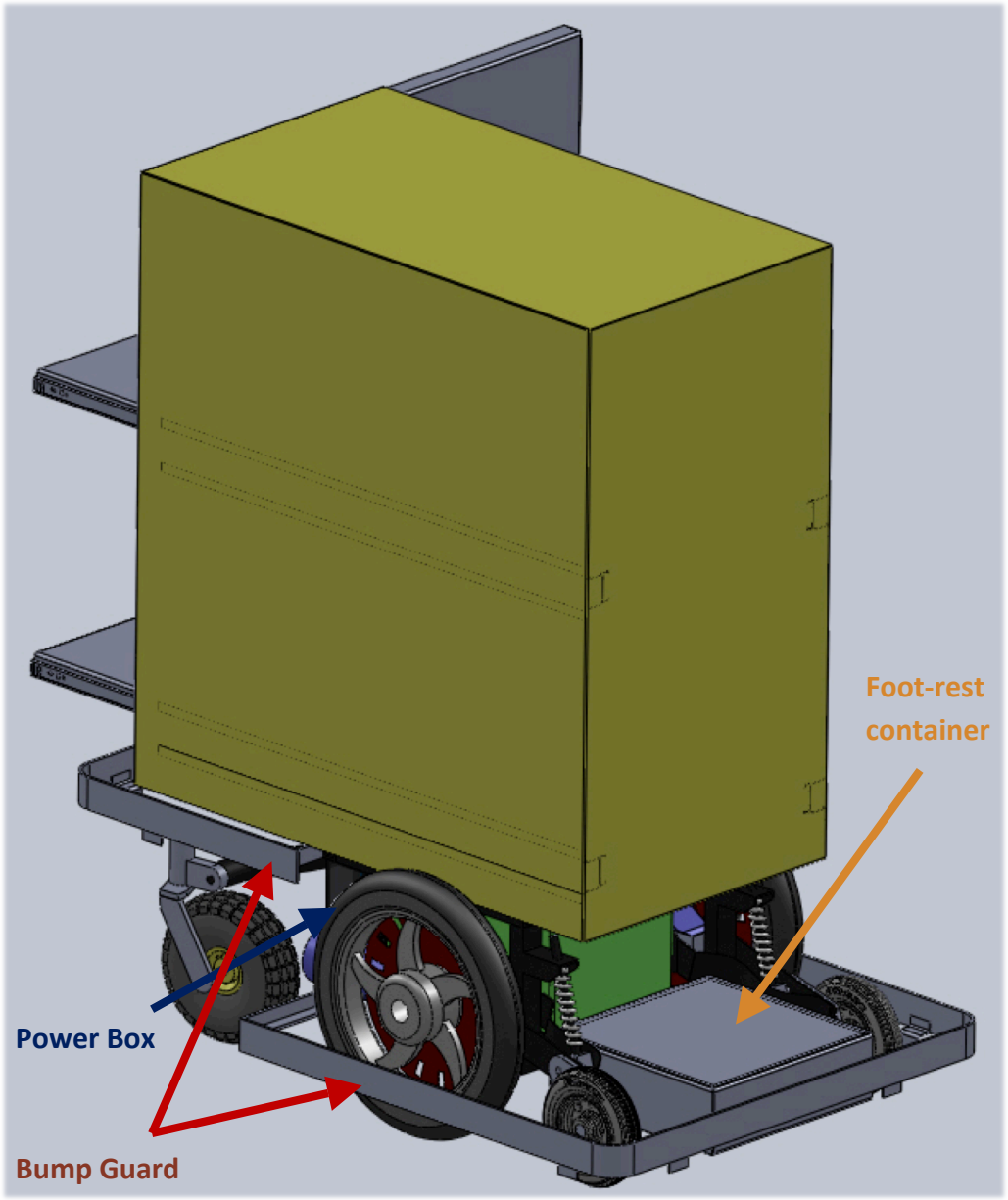


Figure 10 – Final Prototype

SYSTEM OVERVIEW

Figure 11 depicts the entire autonomous system at a high-level. The system level block diagram is broken down into major key areas. The major areas are: user interface, sensors hardware, structure, power, motor control, and system interface. Each team member has a non-overlapping portion of the project to lead as head engineer in charge of research and implementation.

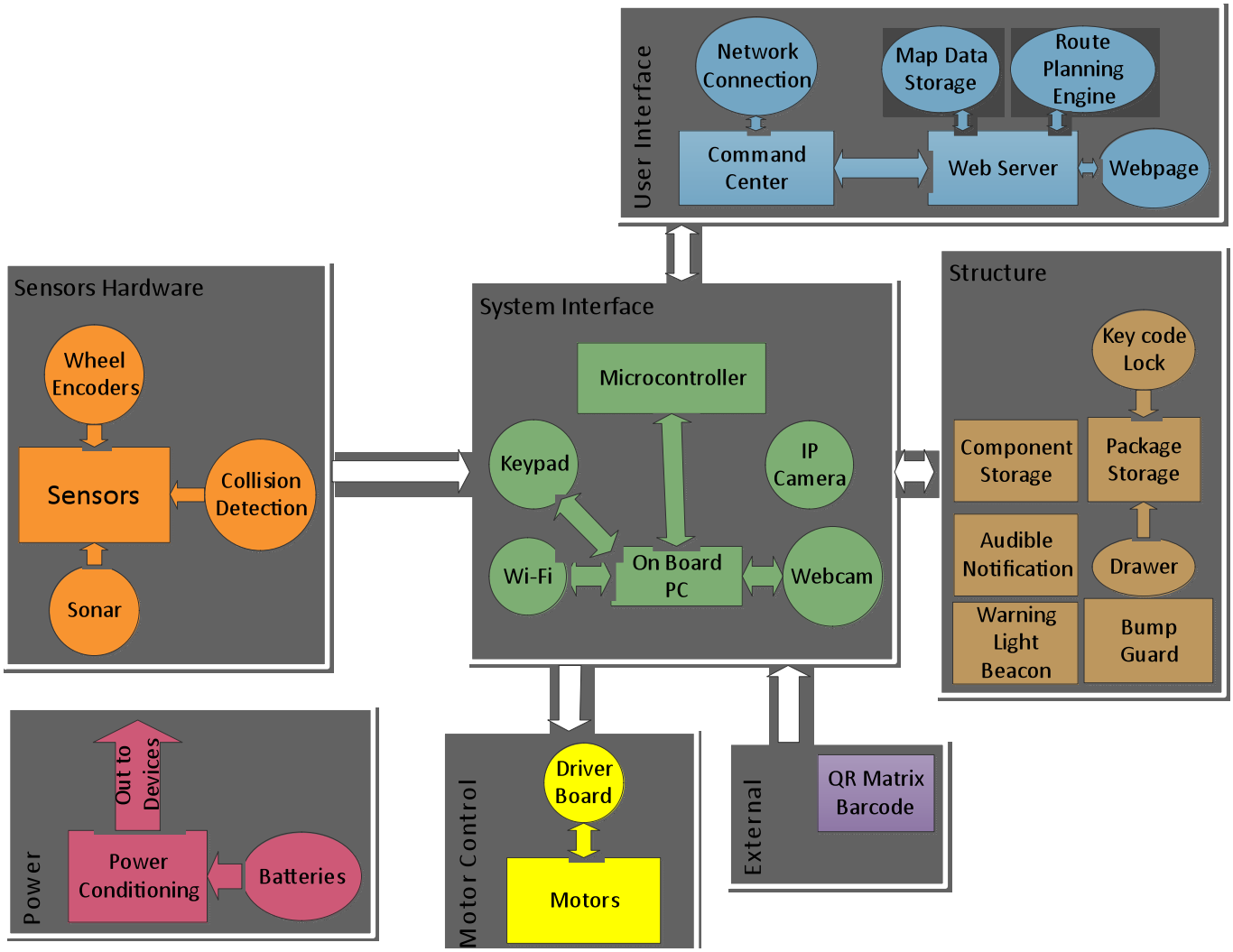


Figure 11- System level block diagram

BUDGET

The budget for this project contains all necessary components to achieve a working prototype. Table 2 is an itemized part and costs list. The prices used were gathered upon what was needed for the project. After discussions with Professor Patrick Mantey from UCSC, we discovered what we should plan on budgeting for.

Itemized Budget

QTY	Price per Item	Item Short Description	Item Subtotal
10	$50.00	Sonar sensor	$500.00
6	$2.00	IR Tape Sensor	$12.00
1	$150.00	Power Components	$150.00
2	$20.00	Encoders	$40.00
1	$275.00	Zotac Computer	$275.00
2	$40.00	Microprocessor	$80.00
1	$90.00	Wireless IP camera	$90.00
1	$300.00	Electric wheelchair	$300.00
1	$75.00	Gel battery charger	$75.00
1	$100.00	PCB Manufacture Run	$100.00
1	$200.00	Metal work (labor and materials)	$200.00
1	$80.00	Discrete components	$80.00
1	$100.00	Cables/wires and connectors	$100.00
1	$200.00	Build material and tools	$200.00
1	$30.00	Safety Beacon	$30.00
1	$30.00	Computer Camera (QR Code Reading)	$30.00
3	$50.00	H-bridge	$150.00
7	$20.00	Bump Sensor Components	$140.00
TOTAL			**$2,262.00**

Table 2 – Preliminary Budget

Our goal is to aim to for an affordable prototype for this type of autonomous system as similar delivery systems, such as those in hospitals, are leased at an approximate price point of $1500 per month,

or $18,000 per year. The initial budget indicates a target cost of just over $2,250 total cost for the basic prototype, a 50% increase to the cost compared to the approximate monthly rental cost of a full featured hospital delivery vehicle[1]. However, our robot is a onetime purchase, which changes the annual cost of our system to .125% compared to the annual cost of the rented system. While adjustments were made throughout the quarter, unexpected cost of components items increased, along with the number items we actually needed. Table 3 shows the actual cost of our project, with the unexpected purchases due to the problems that arose during the project. Comparing this value to the annual cost of a rented delivery robot, we still are looking at a .17% cost compared to that of the annual cost of a rented robot.

Itemized Budget

QTY	Price per Item	Item Short Description	Item Subtotal
20	$38.00	Sonar sensor	$760.00
6	$1.00	IR Tape Sensor	$6.00
1	$160.00	Power Components	$160.00
2	$20.00	Encoders	$40.00
1	$275.00	Zotac Computer	$275.00
2	$35.00	Microprocessor	$70.00
1	$90.00	Wireless IP camera	$90.00
1	$300.00	Electric wheelchair	$300.00
1	$75.00	Gel battery charger	$75.00
1	$150.00	PCB Manufacture Run	$150.00
1	$280.00	Metal work (labor and materials)	$280.00
1	$195.00	Discrete components	$195.00
1	$115.00	Cables/wires and connectors	$115.00
1	$115.00	Build material and tools	$115.00
1	$30.00	Safety Beacon	$30.00
1	$30.00	Computer Camera (QR Code Reading)	$30.00
5	$50.00	H-bridge	$225.00
7	$25.00	Bump Sensor Components	$175.00
1	$30.00	Cooling and Audio	$30.00
TOTAL			**$3,081.00**

Table 3 - Actual Budget

[1] View Aethon's website for more details.

CONCLUSION

The overall project was more complicated than we initially anticipated. Initially, we thought we would reach our goal by getting the robot to move down the hallway on its own, which turned out not to be the case. As we progressed through the project, we found that we needed much more elaborate systems in place to ensure safety, control, and structure.

Safety is what we believe was underestimated. Mainly due to how much time we thought we would need to account for, the safety became complicated. Instead of developing only one safety feature, we developed multiple features to ensure every angle was covered. At first, we were going to only use the sonars to determine whether something was in front of us and stop the robot when an object is within a set threshold range. This developed into something more, such as a mechanical disconnect for the motors, a bump system, and the sonar thresholds.

Control was the next huge part of the system. Within this, we were thinking of minimum control needed, since we wanted to keep it autonomous. This changed a lot, because we found that the intelligence needed to accomplish full autonomy was not as simple as we thought. Instead, as we progressed, we made it controllable through various items. All of the control focuses around the micro, which interprets the data from the PC. This made us create a manual control system, along with an autonomous mode, with multiple different ways to control it. The first way would be to use the existing system on the wheelchair. The second would be a type of remote control done through remote login to the on board PC. Lastly, a keypad connected to the on board PC will also allow for remote control of the bot. All of these work as long as the system is left in manual mode.

Along with control, feedback for the system was extremely underestimated. Fortunately, Professor Gabriel Elkaim was willing to meet with us to discuss our problems. This was scheduled to take

only a week and a half, but turned into an entire month. Unfortunately, all the courses taught at UCSC do not cover the practical applications of feedback control, only the theory behind it. This caused us a lot of problems and was mainly the issue for our delay on this part of the system.

The final piece of our underestimates is the structure itself. Adding all of the components caused us to create multiple custom structures to house the components. It also restricted us to how much we could add to the robot. Ideally, we would want the storage container to be removable, so this left us to design the entire component storage to the frame of the robot. Along with that, we needed to make sure that the components were not exposed. This proved to be more than we expected and caused multiple instances of re-evaluating the layout of where the components needed to be.

As a group, we feel that we have accomplished more than we first set out to accomplish. Part of this was due to our professors and TA, who kept pushing us to do more than we thought we were able to.

FUTURE APPLICATIONS

From the beginning of this project, we felt that this project would be a base model for future projects to build upon. There are many different applications that can be applied to this system, and we feel that we should prove that some of these applications can be done for future projects.

One application in particular is the autonomous use of elevators for the robot to traverse to multiple floors without the aid of people. In most applications, autonomous vehicles cannot move from floor to floor without the assistance of humans. This slightly defeats the purpose of autonomous vehicles in a business environment, since they typically need to go from one floor to another to complete their route. Hospitals are on the cutting edge of this technology, allowing autonomous robots the ability to control the elevator through wireless technology. Companies like Aethon are paving the way for

autonomous robotic delivery systems (ARD), but are not clear on how they use the elevator control system[2].

There is much more research being done on how an autonomous robot can use the elevator. One idea that was done was using image processing to determine the location of elevator buttons[3]. While this system is efficient, it applies more technical abilities of the robot. The system they use determines which button to push based upon image processing, and various effects on the image, in order to find the needed button. However, the robot then implements a physical arm to push the button, mimicking a human pushing the elevator buttons. If this was to be applicable, an attachment would need to be made for our vehicle that will align itself correctly with the button.

Another option for elevator control would involve remote monitoring[4]. The remote monitoring system is used to control the elevator from a remote location and move the robot from floor to floor. However, this is mainly used for security purposes, which means the elevator is probably not interfacing with standard public use. The elevators in our application require the ability to allow people to travel in the elevator with the robot. The remote monitoring system would still work, but there is much more implementation required. The remote monitoring software is specific for only certain models of elevator control systems, and like most new features, cost more. To add this system to our application would require an entirely new elevator control system, and possibly a new elevator system, so this is not an option.

For our system, we have developed a prototype ability to control the elevator. Instead of trying to connect directly to the control box of the elevator, we can mimic a button push electronically. This can

[2] Under Aethon's website, the product information is not detailed enough to determine how the elevator controls work

[3] See Jeong-Gwan, K (2010), Recognition and Path Planning Strategy for Autonomous Navigation in the Elevator Environment, for information on image recognition for elevator controls.

[4] See Shimosasa, Y. (1999), Security service systems using autonomous mobile robot, page 828 for remote monitoring of robots.

be done by using a Bluetooth device that acts like a wireless serial connection. This connection will be made on a microcontroller that will interpret the commands sent through the Bluetooth to electronically push a button. This control system will be next to the push button on the outside of the elevator, simulating a call to which direction the robot will need to go, i.e. up or down. Another set-up, identical to the outer system will be implemented within the elevator car. This system will control where the robot needs to go by selecting the correct floor. Something to consider for this portion is multiple floors selected and how the robot will need to monitor which floor it is on so it does not leave the elevator car on the wrong floor. Fortunately, this can be done on microcontroller by sending a signal back to the robot of the current floor the car is on.

The unfortunate side effect with choosing Bluetooth for the communications is the limited number of paired items, usually around seven pairs at a time. This would limit our robot to one car, and only six floors. Ideally, we would want to able to scale this number to any number of floors needed. For this purpose, we should choose a radio communication that can implement a larger number of paired devices, or channel control each floor selection, such as Xbee, or Zigbee.

Another application that can be built upon is the use of the QR codes as an assistive technology. Since the robot is essentially blind, then a blind person can also use the QR codes to discover where in the building they are. A device would need to be made that would read the QR codes when they were close, like a cellular phone, or camera on a pair of sunglasses. This camera could read the QR code, like our system does, but initiate an audible device that tells the person where they are, where to go next, or if they have gone the wrong direction.

Other applications can be initiated into this system, such as building to building travel, closer quarter algorithms (for elevators with people in them and doorways), along with line-up algorithms for going through elevator doors and doorways without triggering the bump sensors, or getting caught in the

doorway. This may also require a mechanism, or even wireless control for opening the doorways for as long as needed for the robot to get through.

WORKS CITED

Aethon. (2008). Automated Hospital Delivery and Asset Management Solutions, from http://www.aethon.com/products/logistics.php

Bhuiyan, M. Z. H., Lohan, E. S., & Renfors, M. (2010, 30 May-2 June). A Slope-Based Multipath Estimation Technique for Mitigating Short-Delay Multipath in GNSS Receivers. Paper presented at the 2010 IEEE International Symposium on Circuits and Systems. ISCAS 2010, Paris, France.

Jeong-Gwan, K., Su-Yong, A., Won-Seok, C., & Se-Young, O. (2010). Recognition and Path Planning Strategy for Autonomous Navigation in the Elevator Environment. International Journal of Control, Automation, and Systems, 8(4), 808-821. doi: 10.1007/s12555-010-0413-3

Jia, P., & Xianhai, Y. (2010). A Design of an Autonomous Mobile Robot's Navigation System Based Behaviours. Katsuki, R., Ota, J., Tamura, Y., Mizuta, T., Kito, T., Arai, T., . . . Nishiyama, T. (2003, 27-31 October). Handling of objects with marks by a robot. Paper presented at the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, NV,.

Kawewong, A., Tongprasit, N., Tangruamsub, S., & Hasegawa, O. (2011). Online and Incremental Appearance-based SLAM in Highly Dynamic Environments. [Article]. International Journal of Robotics Research, 30(1), 33-55. doi: 10.1177/0278364910371855

Klingbeil, E. (2010). Autonomous operation of novel elevators for robot navigation. In B. Carpenter, Russakovsky, O., Ng, A.Y. (Ed.), (pp. 751).

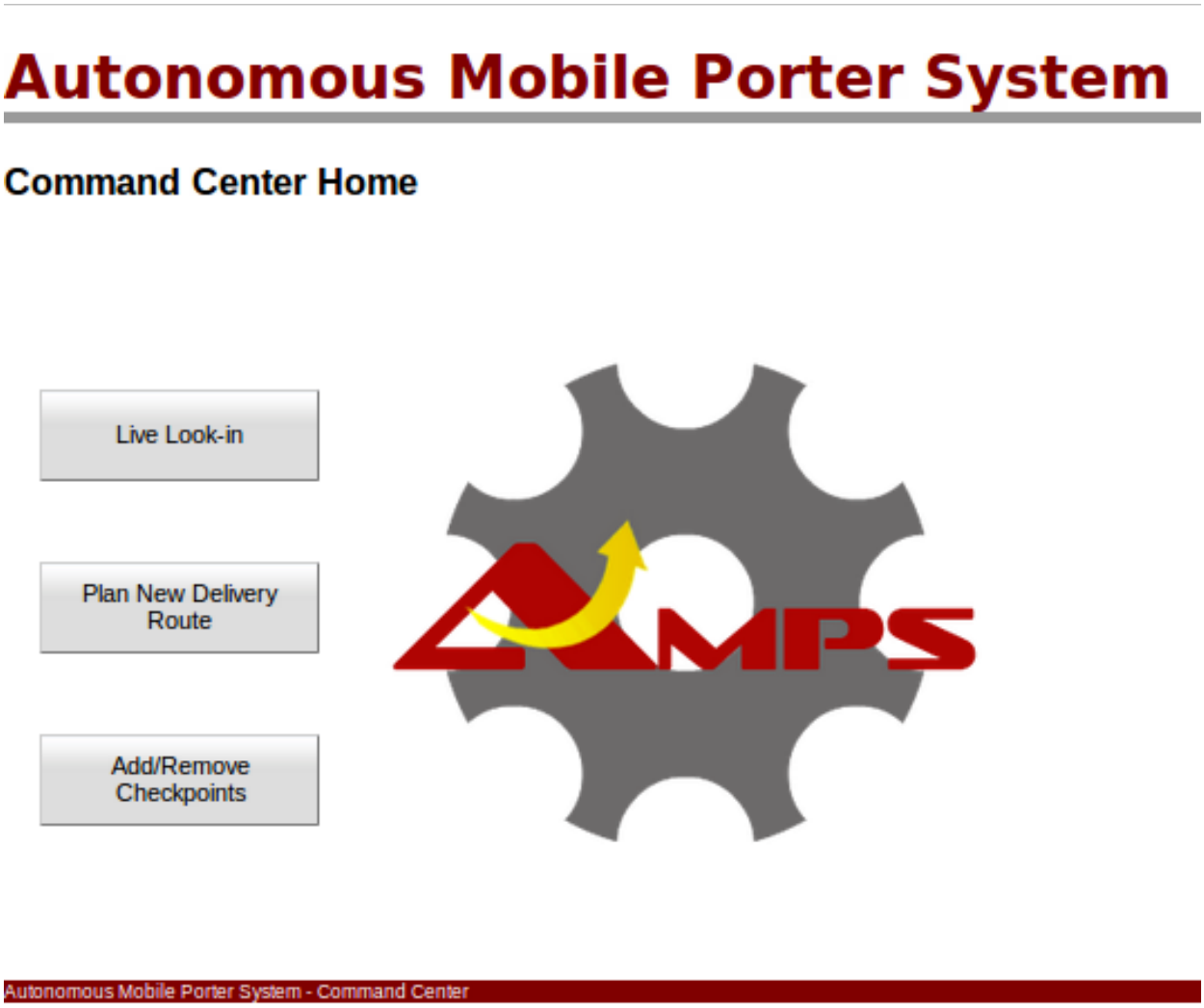
Roy, P. P., Pal, U., & Llado s, J. (2011). Document seal detection using GHT and character proximity graphs. Pattern Recognition, 44(6), 1282-1295. doi: 10.1016/j.patcog.2010.12.004

Shimosasa, Y. (1999). Security service system using autonomous mobile robot. In J. Kanemoto, K. Hakamada, H. Horii, T. Ariki, Y. Sugawara, F. Kojio, A. Kimura & S. Yuta (Eds.), (Vol. 4, pp. 825-829): Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on , vol.4, no.

Xinyue, Z., Satoh, Y., Takauji, H., Kaneko, S., Iwata, K., & Ozaki, R. (2011). Object detection based on a robust and accurate statistical multi-point-pair model. Pattern Recognition, 44(6), 1296-1311. doi: 10.1016/j.patcog.2010.11.022

APPENDIX A: GRAPHICAL USER INTERFACE

HOME PAGE



PLAN NEW DELIVERY ROUTE PART 1

Route Planner

Directions:
Below is a list of all possible destinations (as assigned when checkpoint was added to floorplan).
To create a route follow these steps:

1. Simply click the **"Add To Route"** button to the right of the desired checkpoint location to add destination to route.
   - You may add or remove any amount of destinations per route.
2. Once you have selected all destinations, click **"Continue"** to proceed to start location selection

QR Code	*Bldg. Name*	*Bldg. Floor*	*Location Description*	*Location POC*	
BE:2:250	Baskin	2			Add To Route
BE:2:258	Baskin	2			Add To Route
BE:2:262	Baskin	2			Add To Route

Back to Main Continue

Autonomous Mobile Porter System - Command Center

PLAN NEW DELIVERY ROUTE PART 2

Choose Your Starting Point

QR Code	*Bldg. Name*	*Bldg. Floor*	*Location Description*	*Location POC*	
BE:2:FREIGHT	Baskin	2			Select

Back to Main

Autonomous Mobile Porter System - Command Center

ADDING AND REMOVING CHECKPOINTS

Add/Remove Checkpoints

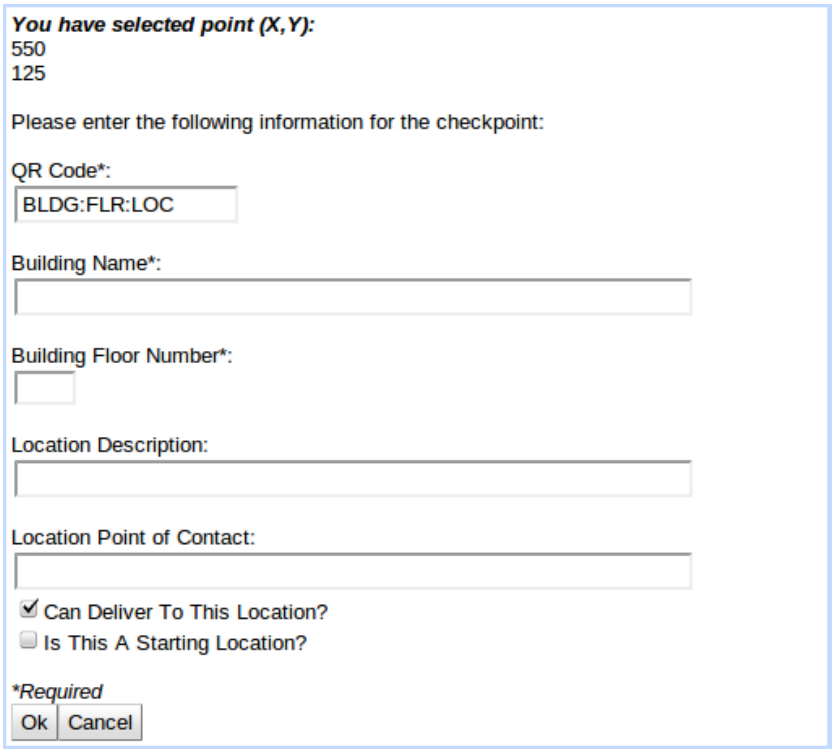
Directions:

- To add a checkpoint
  1. Click inside the green area.
  2. Wait for checkpoint info pop-up.
  3. Fill in the pop-up fields for the new checkpoint.
- To remove a checkpoint
  1. Click any existing map pin.
  2. The information pop-up will display.
  3. To remove, click **"Delete"**
- Checkpoints may be placed anywhere in the green area.
- An existing checkpoint is indicated by a red map pin.
- QR codes may not be duplicated. If you add a point usingthe name of an existing checkpoint, the old checkpoint will be removed automatically before adding the new point.
- *NOTE: In order for a point to be reachable from anotherpoint, there must be a line-of-sight between the two points. Please keep this in mind when installing points.*

Back to Main

Autonomous Mobile Porter System - Command Center

CHECKPOINT ADDITION



CURRENT CHECKPOINT

You have selected the point at (X,Y):
650
125

QR Code:
BE:2:I1

Building Name:
Baskin

Building Floor Number:
2

Location Description:

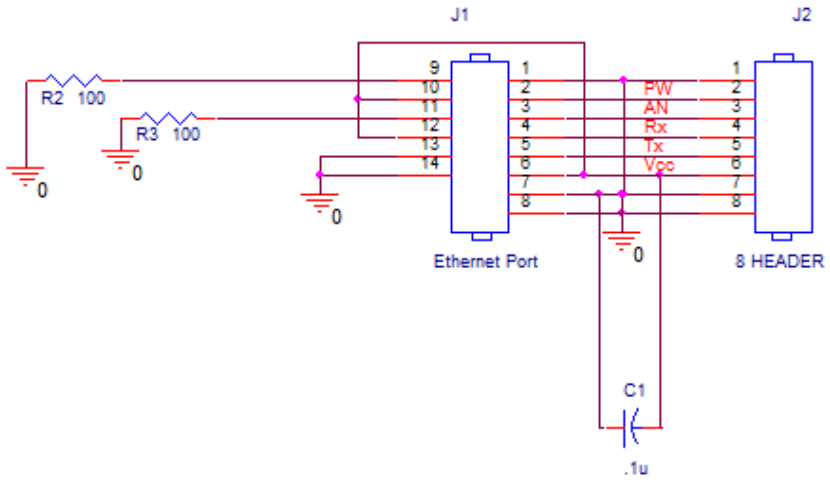
Location Point of Contact:

Can Deliver To This Location? false
Is This A Starting Location? false
Delete Cancel

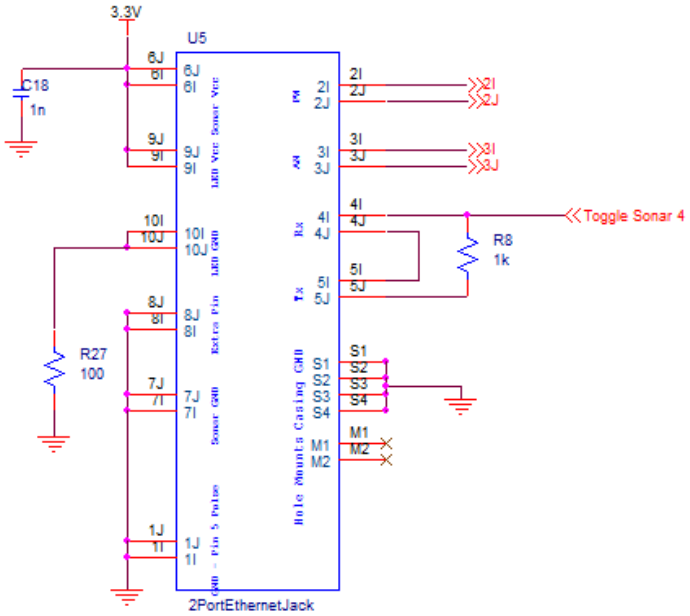
APPENDIX B: ELECTRICAL SCHMATICS

SONAR SENSORS

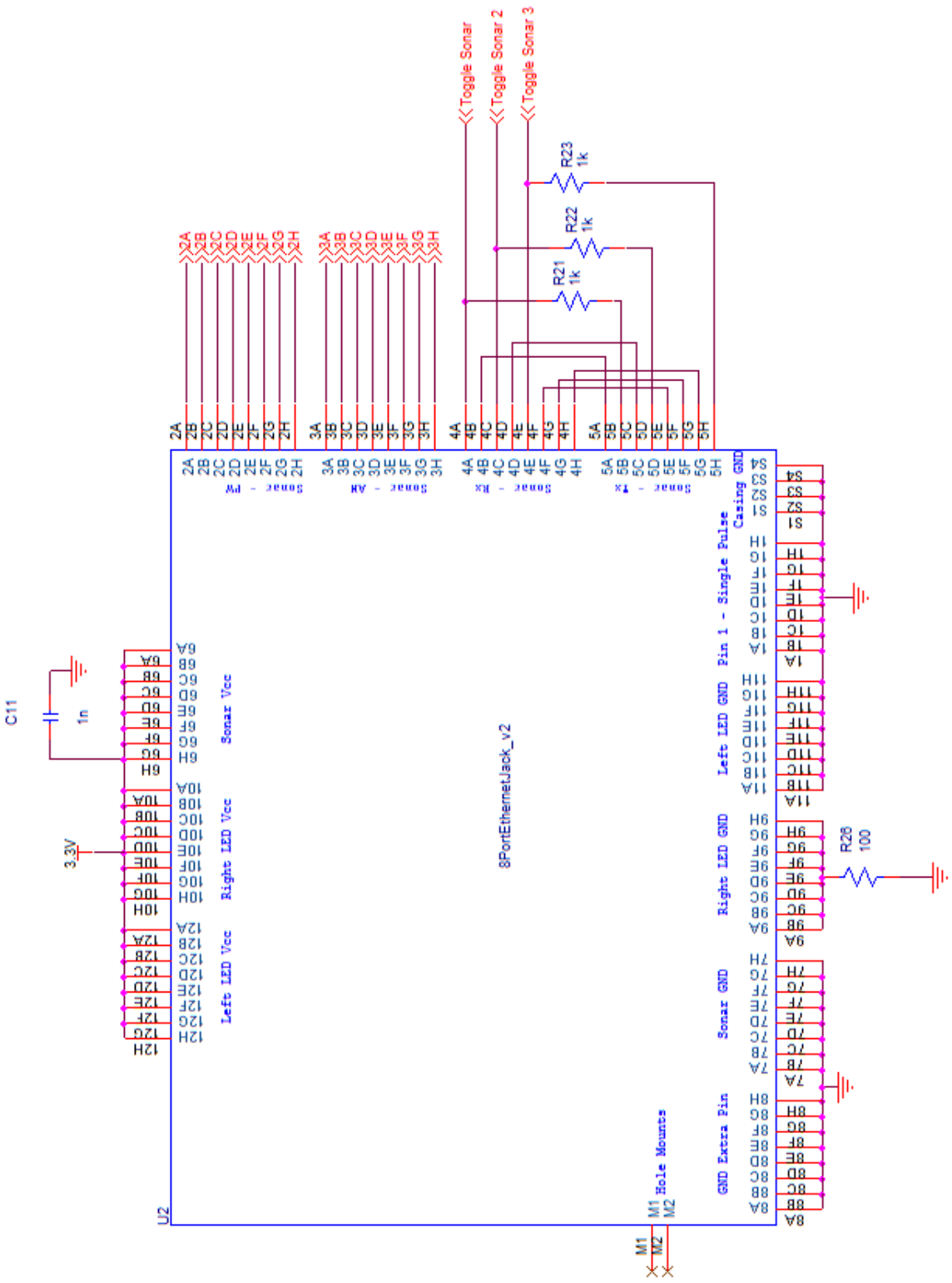
DAUGHTER BOARD



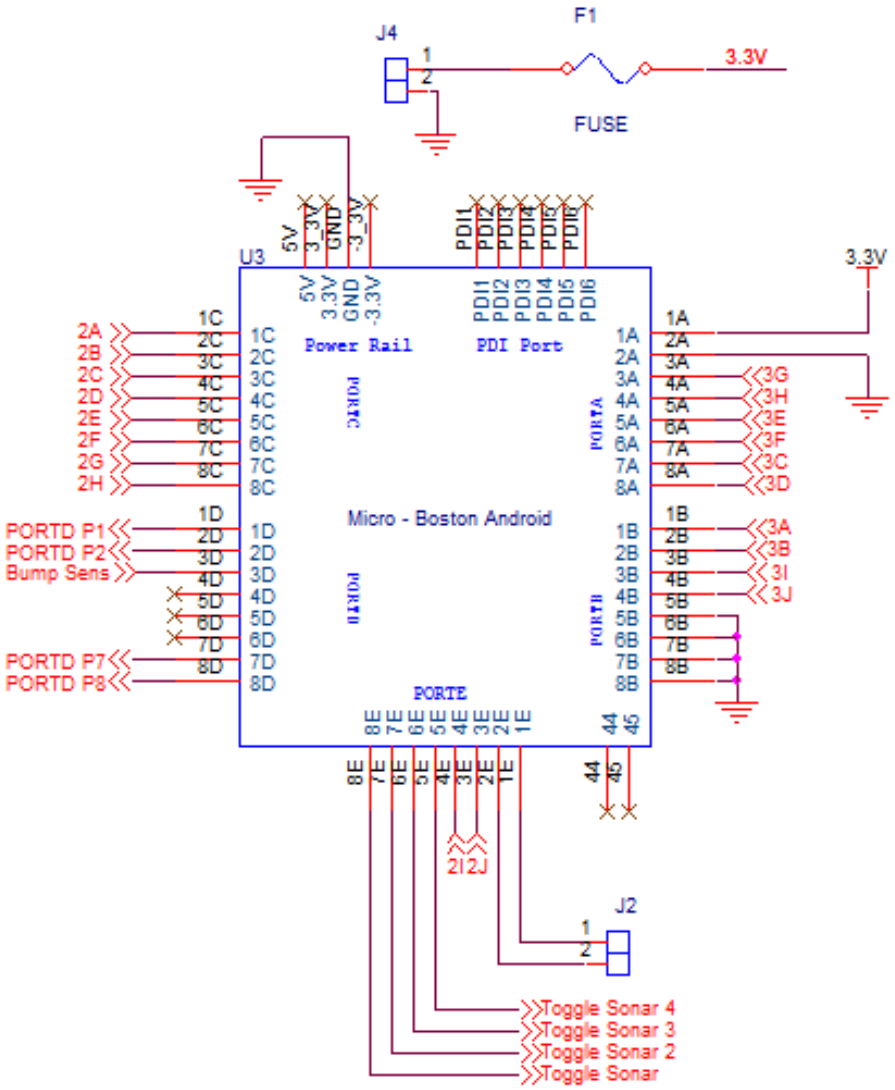
2 PORT ETHERNET CONVERSION



8 PORT ETHERNET CONVERSION

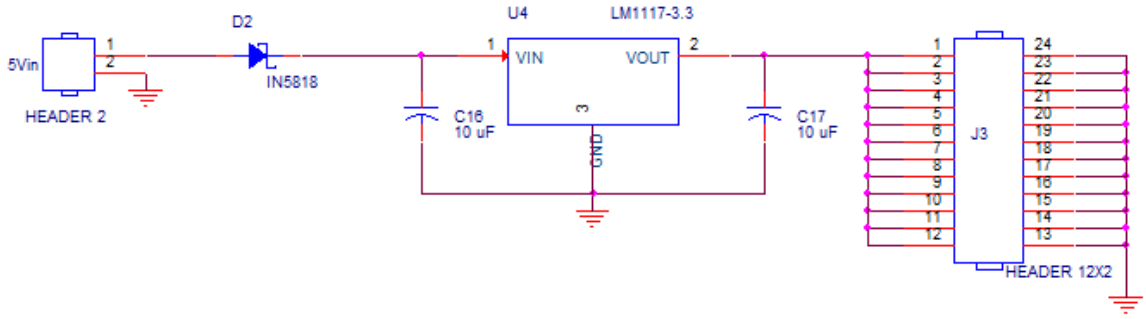


MICROCONTROLLER CONNECTION

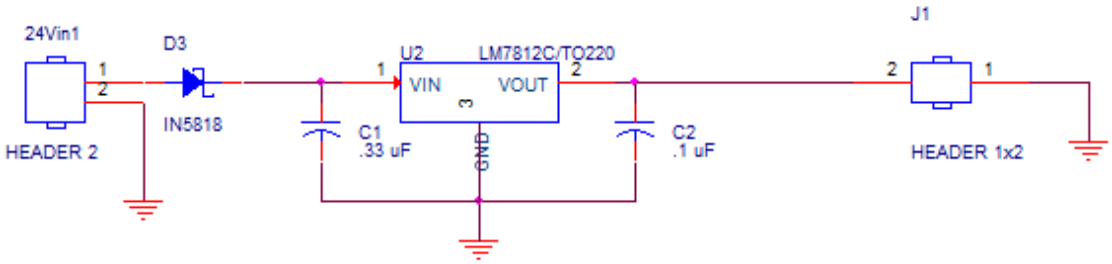

J4
F1
3.3V
FUSE
U3
5V
3.3V
GND
-3.3V
PDI1
PDI2
PDI3
PDI4
PDI5
PDI6
Power Rail
PDI Port
Micro - Boston Android
PORTC
PORTA
PORTB
PORTE
PORTD P1
PORTD P2
Bump Sens
PORTD P7
PORTD P8
Toggle Sonar 4
Toggle Sonar 3
Toggle Sonar 2
Toggle Sonar
J2

POWER RAILS

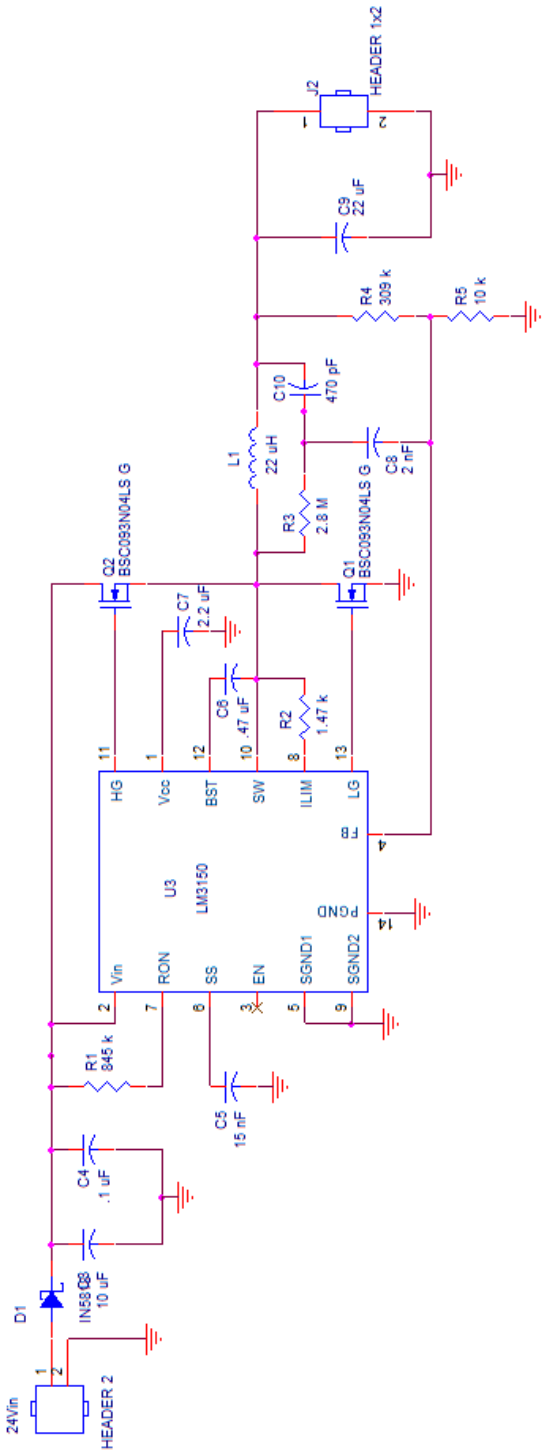
3.3V RAIL



12V RAIL

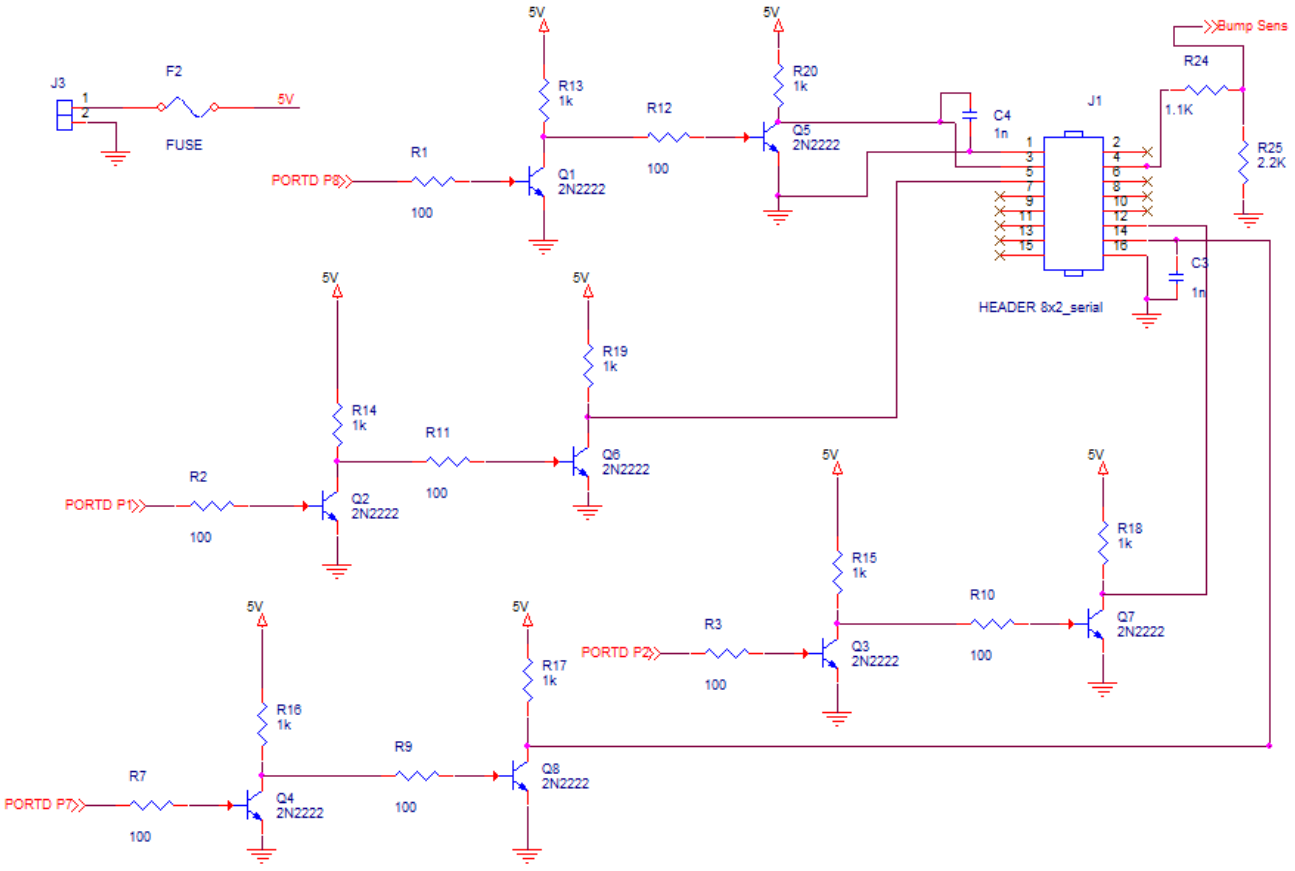


19V RAIL

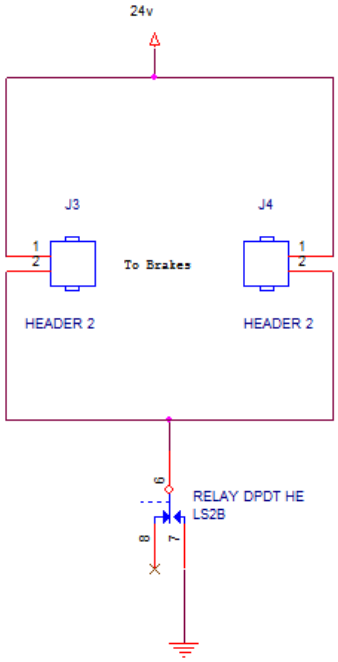


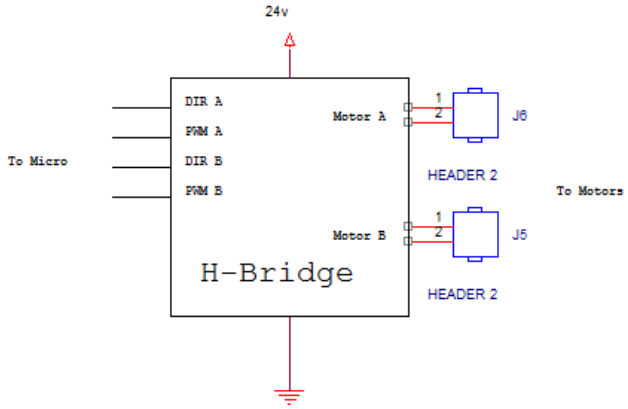
MOTOR SYSTEM

H-BRIDGE FOR PCB

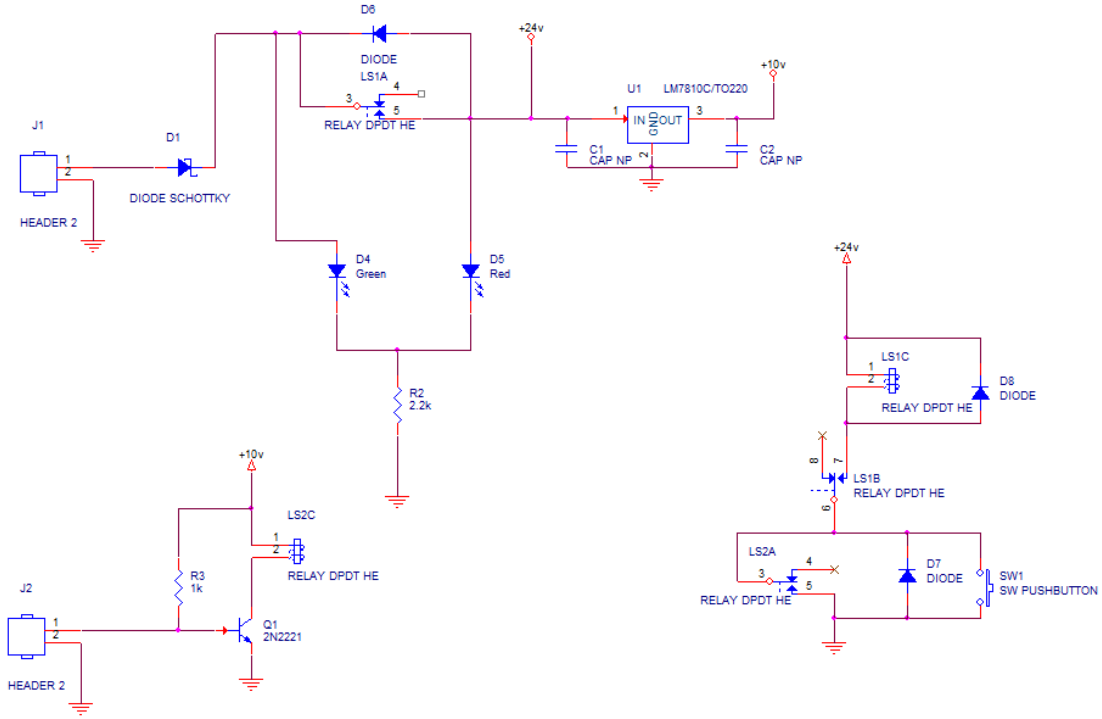


H-BRIDGE FOR PROTOTYPE BOARD

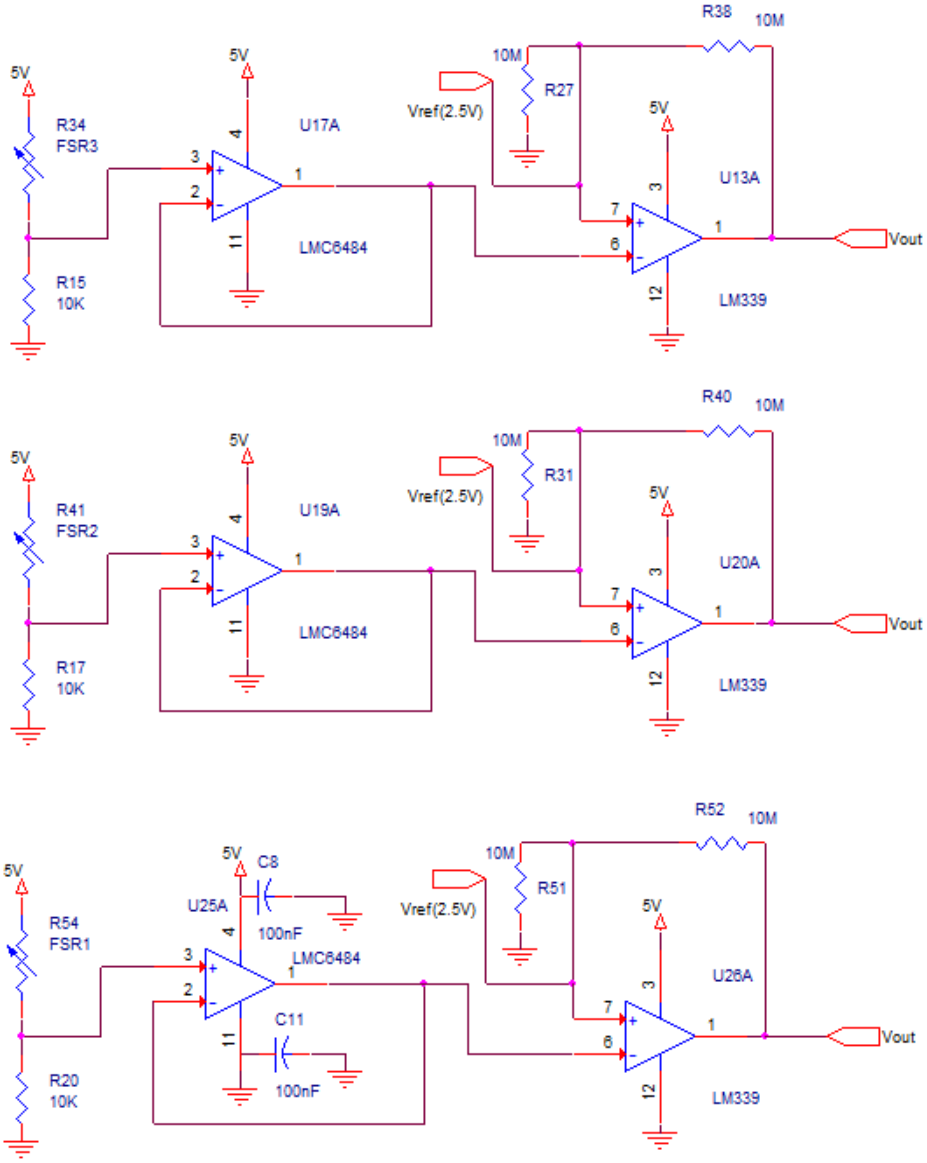


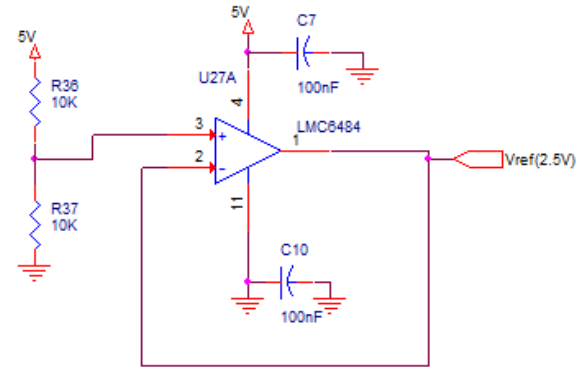


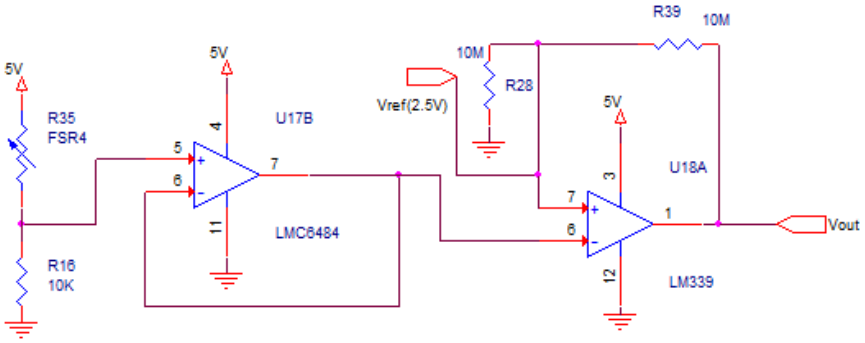
DRIVER BOARD

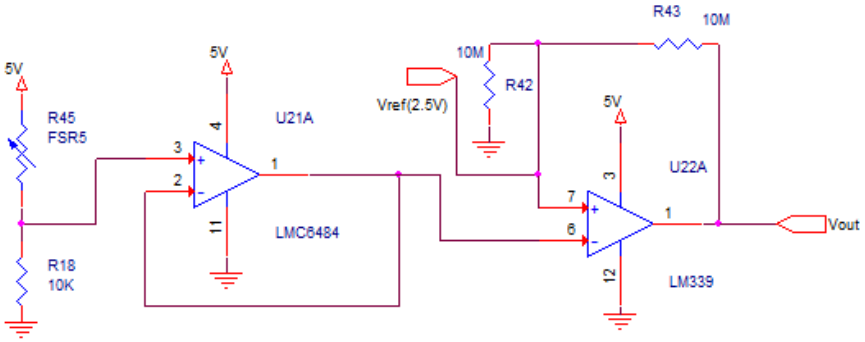


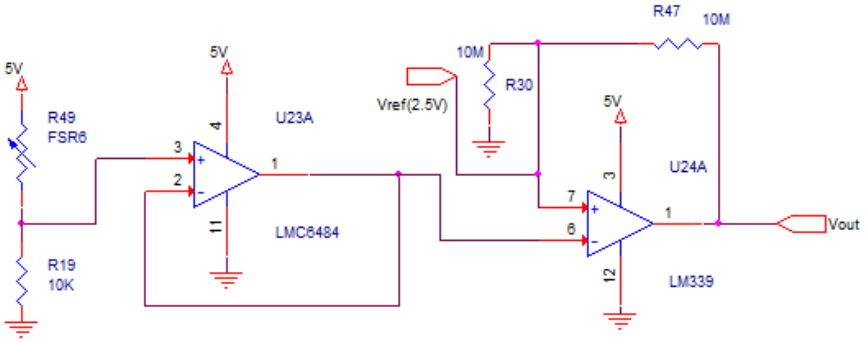
BUMP SYSTEM


R38
10M
5V
10M
R27
Vref(2.5V)
5V
U13A
R34
FSR3
U17A
LMC6484
Vout
R15
10K
LM339
R40
10M
10M
R31
Vref(2.5V)
5V
U20A
R41
FSR2
U19A
LMC6484
Vout
R17
10K
LM339
R52
10M
10M
R51
Vref(2.5V)
5V
U26A
C8
100nF
U25A
R54
FSR1
LMC6484
C11
100nF
Vout
R20
10K
LM339

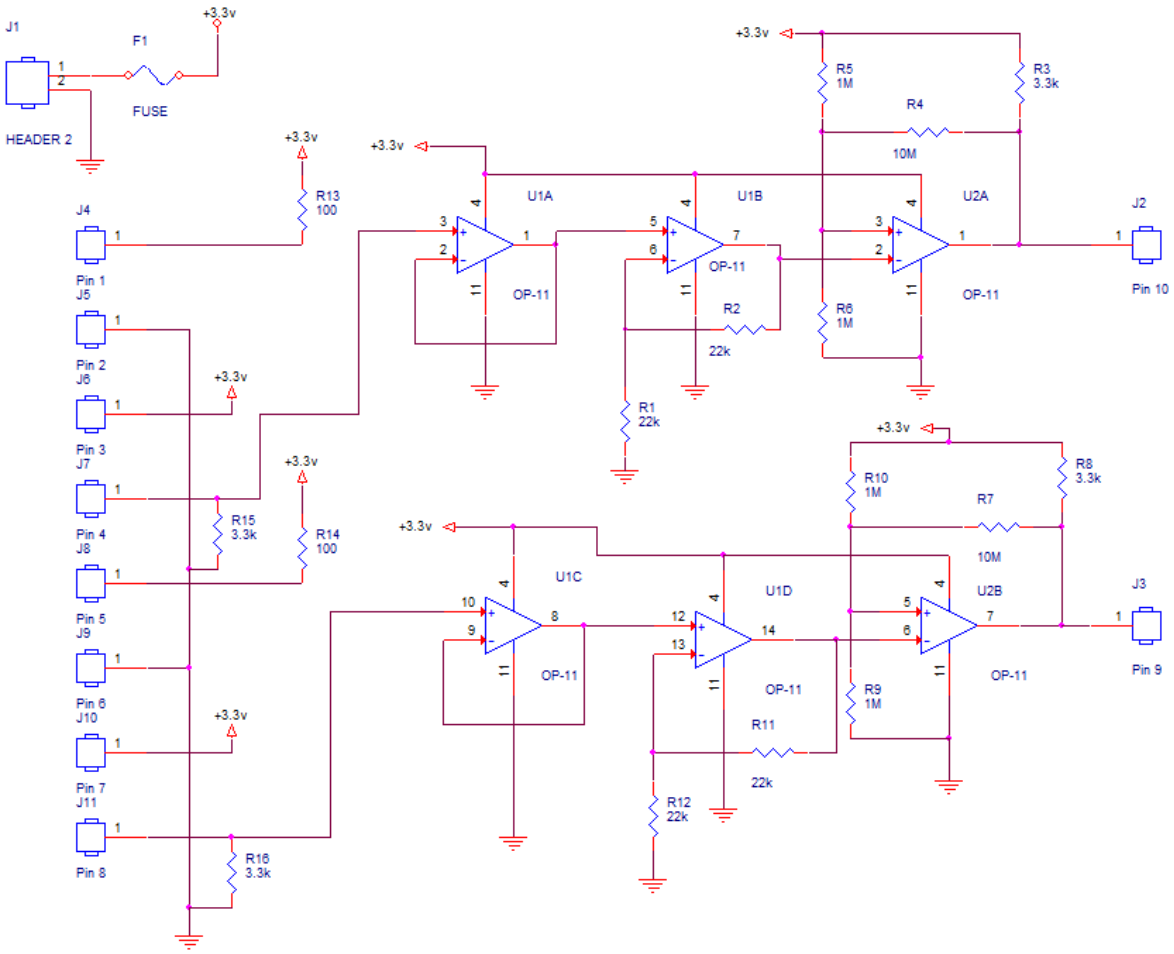

5V
C7
100nF
U27A
R36
10K
LMC6484
Vref(2.5V)
R37
10K
C10
100nF


R39
10M
10M
R28
Vref(2.5V)
5V
U17B
U18A
R35
FSR4
LMC6484
LM339
Vout
R16
10K


R43
10M
10M
R42
Vref(2.5V)
5V
U21A
U22A
R45
FSR5
LMC6484
LM339
Vout
R18
10K


R47
10M
10M
R30
Vref(2.5V)
5V
U23A
U24A
R49
FSR6
LMC6484
LM339
Vout
R19
10K

ENCODERS


J1
F1
+3.3v
FUSE
HEADER 2
J4
Pin 1
J5
Pin 2
J6
Pin 3
J7
Pin 4
J8
Pin 5
J9
Pin 6
J10
Pin 7
J11
Pin 8
R13
100
R14
100
R15
3.3k
R16
3.3k
U1A
U1B
U1C
U1D
U2A
U2B
OP-11
R1
22k
R2
22k
R3
3.3k
R4
10M
R5
1M
R6
1M
R7
10M
R8
3.3k
R9
1M
R10
1M
R11
22k
R12
22k
J2
Pin 10
J3
Pin 9

APPENDIX C: MECHANICAL PROTOTYPES

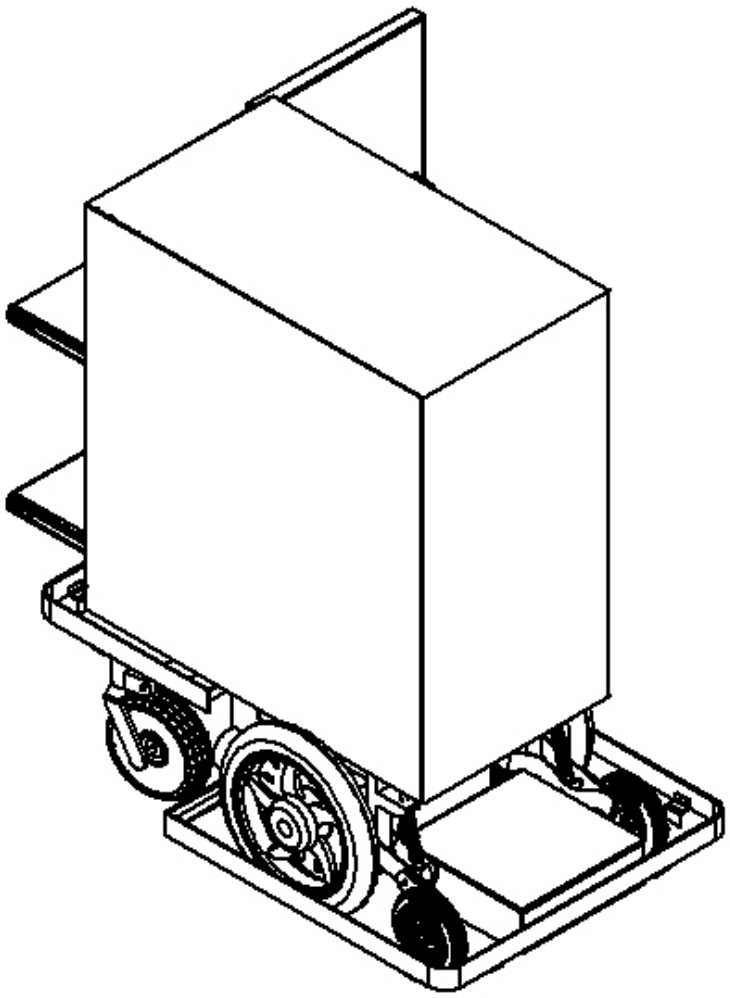


Figure C1 - Prototype (isometric)

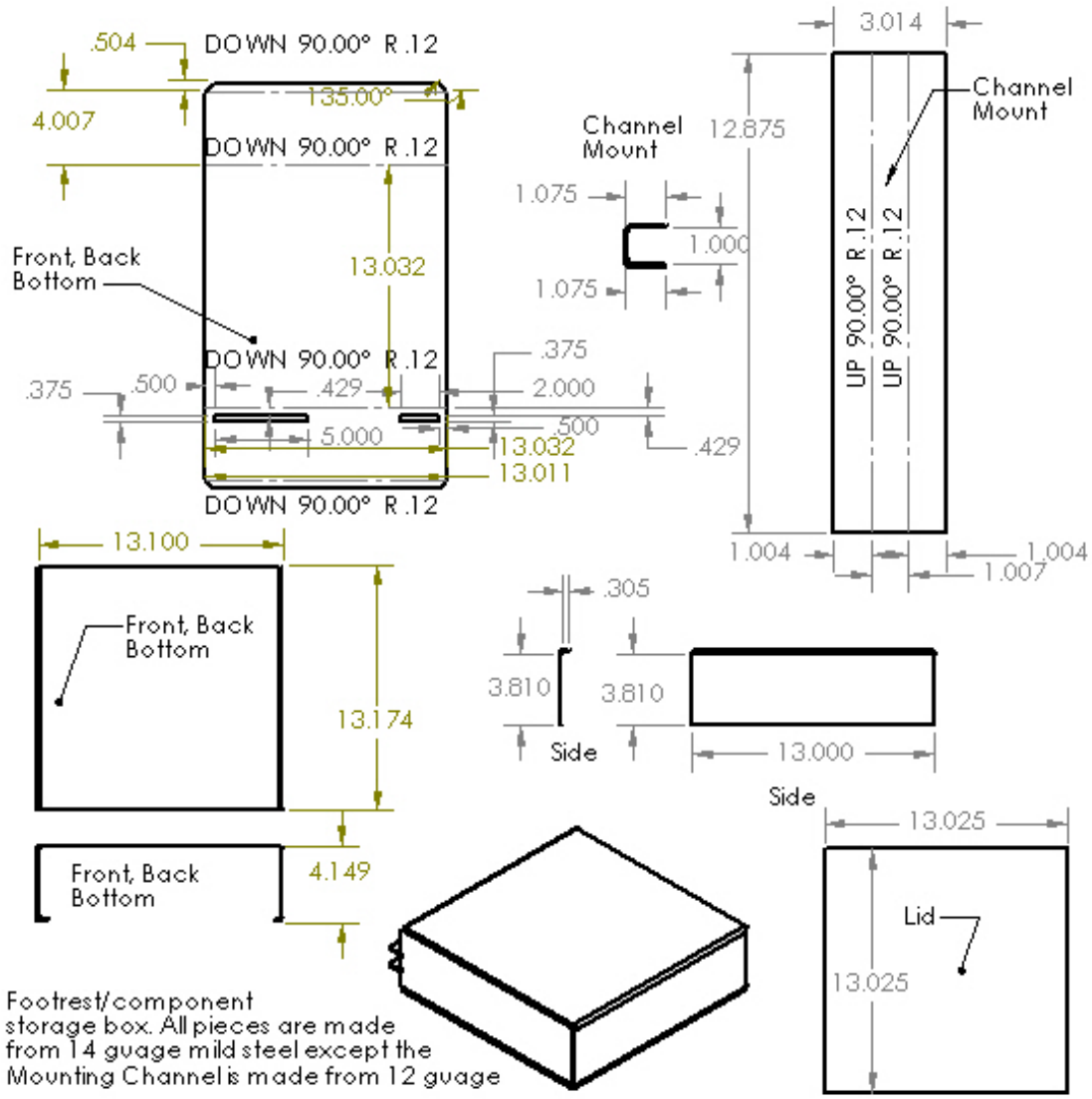


Figure C2 – Component Box

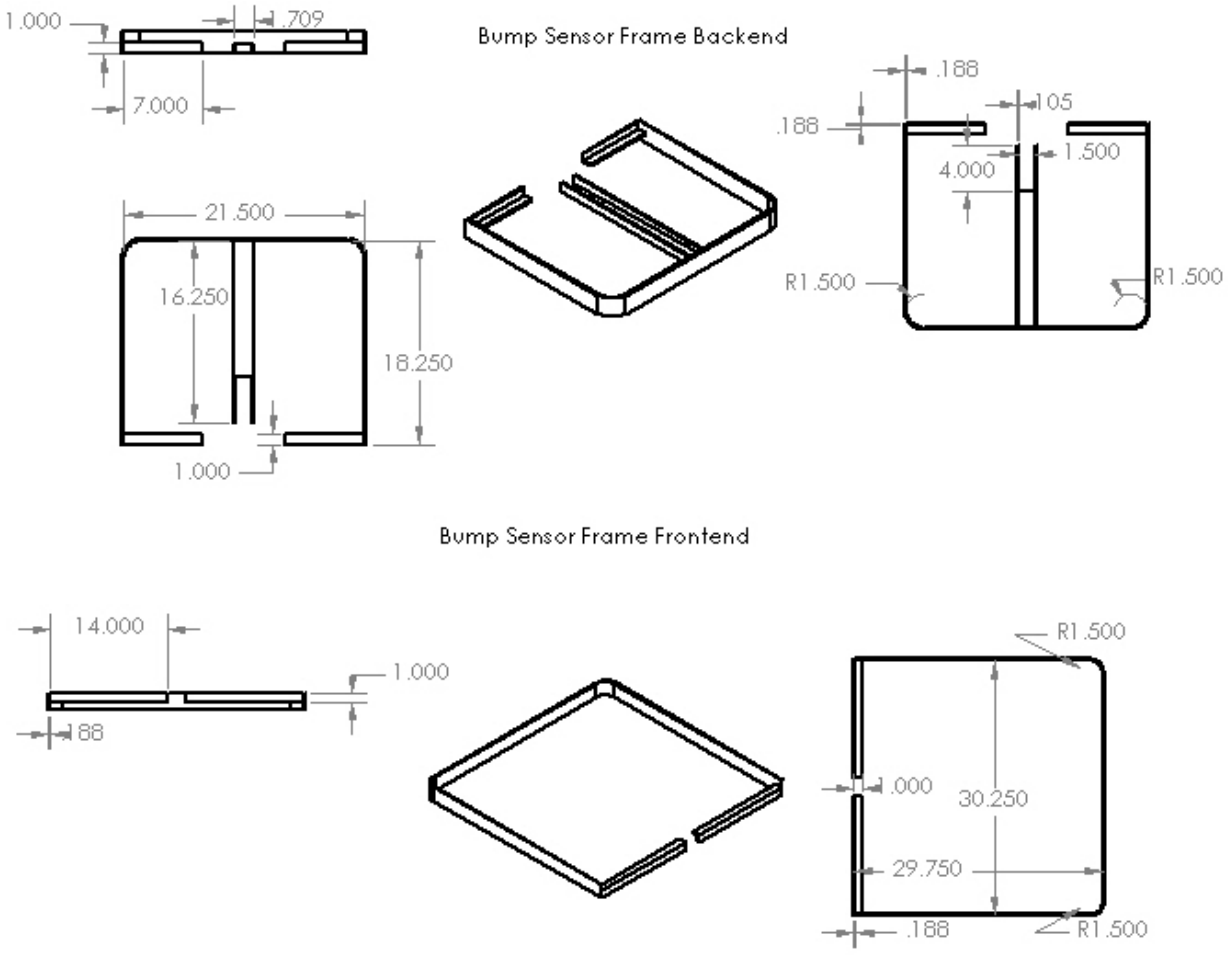


Figure C3 – Bump Guard

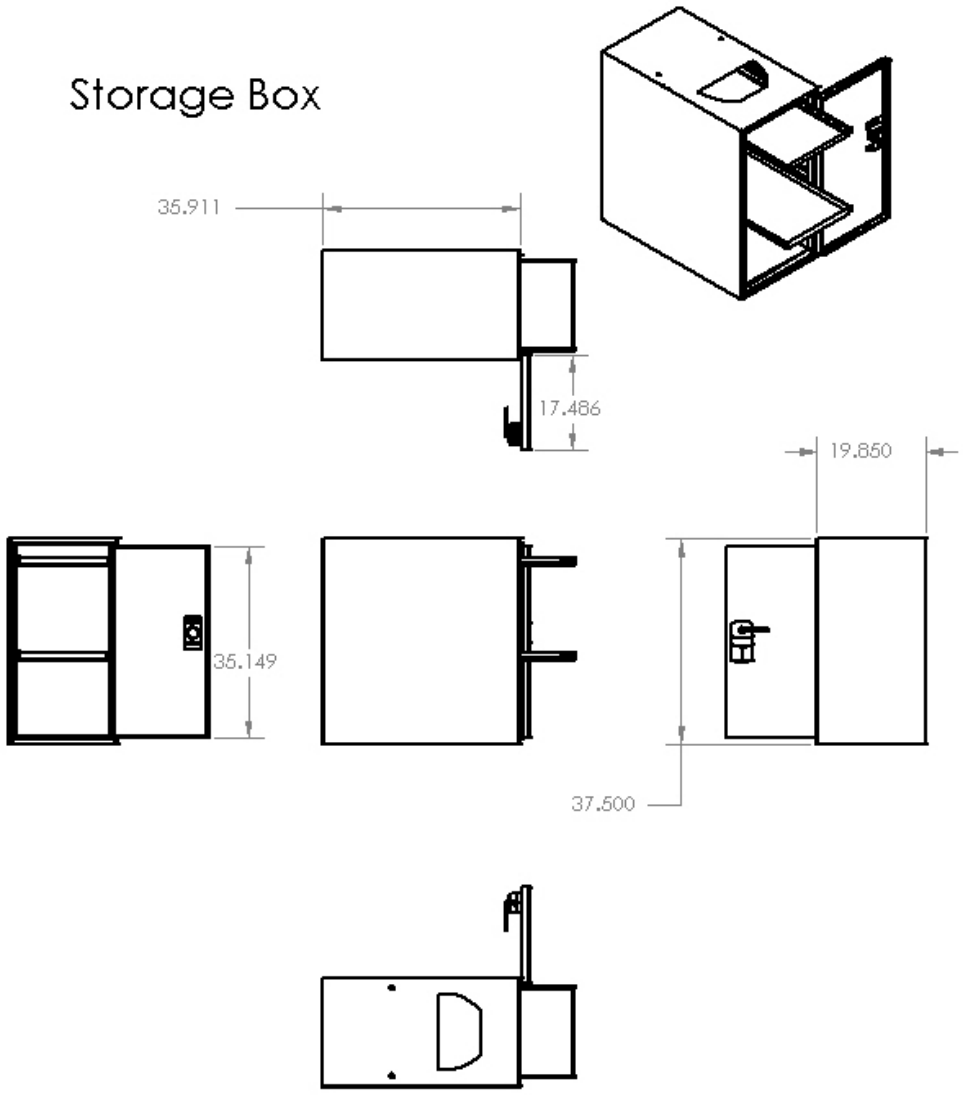


Figure C4 – storage box

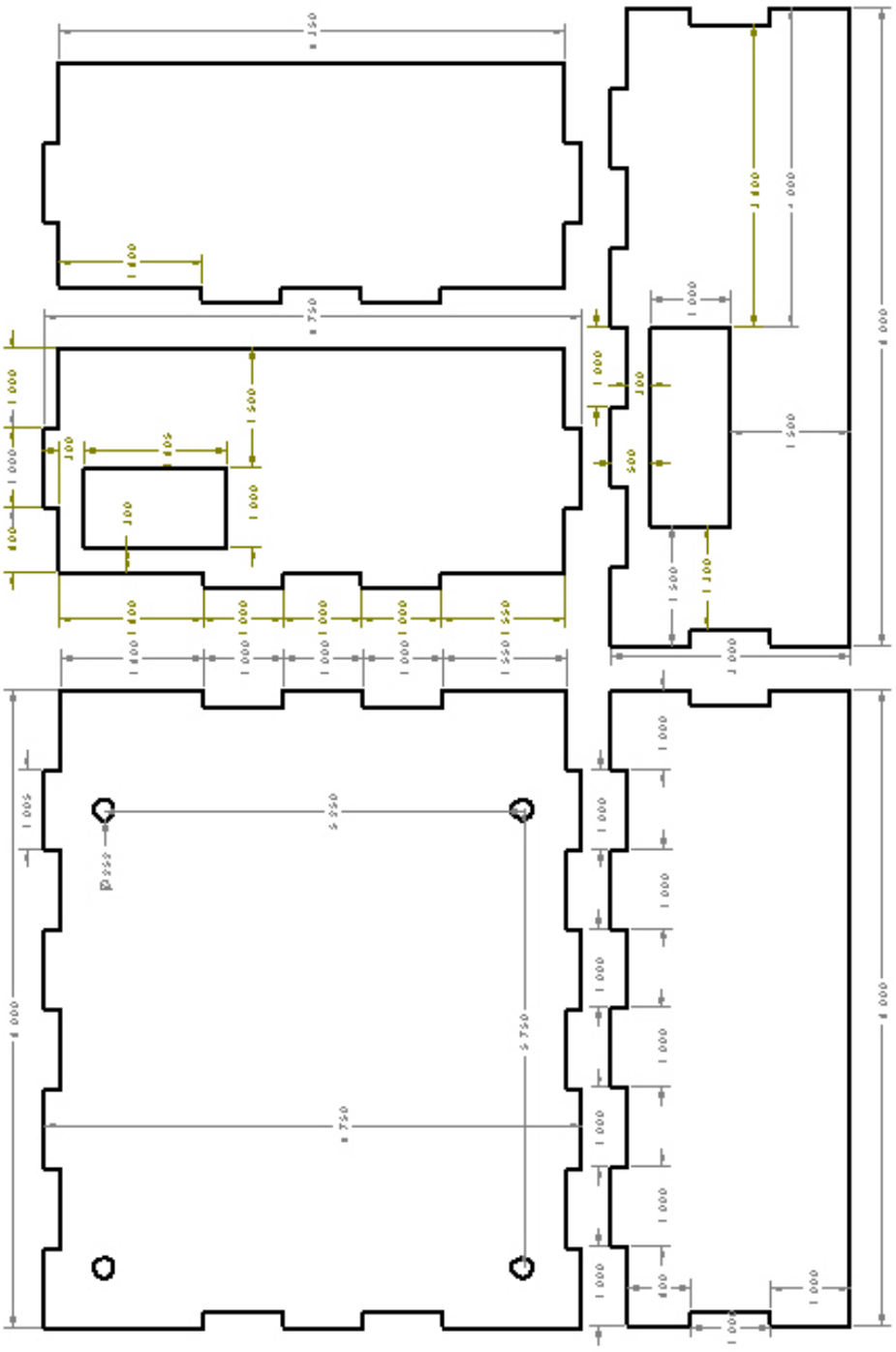


Figure C5 powerbox

APPENDIX C: TEAM CHARTER

CONTACT INFORMATION

TEAM E-MAIL
amps-contact@gforge.soe.ucsc.edu

BRADY BOONE
Bioelectronics and Computer Engineer
Parts Manager
bboone@ucsc.edu
bradylynnboone@gmail.com
(209) 505-2528

ALEJANDRO CERDA
Computer Science
Documentation Administrator
acerda@ucsc.edu
alejandrocerdajr@gmail.com
(415) 341-2253

NOLAN LAU
Computer Engineer
Internal Affairs Officer
nxlau@ucsc.edu
nolanl824@gmail.com
(415) 806-7885

PAUL NAUD
Electrical Engineer
Financial Officer
pnaud@soe.ucsc.edu
pauln36@yahoo.com
(209) 256-1821

CHASEN PETERS
Electrical Engineer
AMPS Team Lead
cpeters@ucsc.edu
chasenpeters@gmail.com

(707) 853-3351

OVERVIEW

MISSION STATEMENT

To successfully design and build an autonomous vehicle capable of intelligently delivering payload to a specified location within a structure.

PROJECT DESCRIPTION

The purpose of this project is to build an autonomous vehicle that will deliver a payload to a corresponding location within a building, such as Jack Baskin Engineering. This will be accomplished by training the vehicle to memorize its surrounding environment using an array of onboard sensors and central processing unit. The map data collected is transmitted to the command center via wireless communication. The command center, a dedicated PC running custom software, is responsible for collecting, managing, and processing map data for both route planning and end-user status information. It will also serve as the end-user interface for loading payload information onto the vehicle. The calculated route is then transmitted to the vehicle and activates the start of the delivery process. The vehicle will intelligently navigate the predetermined route via an X and Y position tracking system coupled with location checkpoints. Once the vehicle reaches a destination, it will send notification of its arrival to the command center. Once tasks are accomplished at the destination, the vehicle proceeds on route until all payload items have been delivered.

DIVISION OF LABOR

Every member of the team must hold the position of lead engineering for at least one (or portion of) major component block. They will have the responsibility of managing all tasks associated with their assigned block to ensure all actions are completed correctly, in line with project specifications, and in accordance with the project schedule and standards. The major component blocks and assignments are assigned as follows:

- *Brady Boone:*
  Will serve as lead engineer for **vehicle mechanics and structure**.
  As lead structural and mechanical engineer, he will be required to oversee work involved with modifications, additions, and/or design of all mechanical components that will reside on the autonomous vehicle. Components include, but are not limited to, the vehicle chassis, wheels, payload storage mechanism, and electrical hardware component mounting.
  He has been assigned the task of backup lead engineer for sensors hardware (sonar and encoders) to assist Nolan Lau if so determined by team lead.
  In addition, he will also serve as the Parts Manager in charge of ensuring accurate ordering and tracking of all project parts and electrical components.

- *Alejandro Cerda:*
  Will serve as lead engineer for the **command center and software.**
  As lead software engineer, he will be required to oversee work involved with design, implementation, and deployment of the command center software to ensure all route planning, communication protocols, and user interface are in correct working order.
  He has been assigned the task of backup lead engineer for the onboard processing unit and motor control to assist Chasen Peters if so determined by team lead.
  In addition, he will also serve as the Documentation Administrator and assistant project lead. This includes, but is not limited to, managing all project documentation, updating project metrics to maintain accurate record keeping, and managing project schedule.

- *Nolan Lau:*
  Will serve as lead engineer for **sensors hardware (sonar, encoders, and IR)**.
  As lead sensors hardware engineer, he will be required to oversee work involved with the design, implementation, and interfacing of the sonar, encoder, and infrared tape array sensor suite.
  He has been assigned the task of backup lead engineer for command center and software to assist Alejandro Cerda if so determined by team lead.
  In addition, he will also serve as the Internal Affairs Officer in charge of facilitating member interactions.

- *Paul Naud:*
  Will serve as lead engineer for **sensors hardware (RFID) and power distribution**.
  As lead sensors hardware engineer, he will be required to oversee work involved with the design, implementation, and interfacing of the radio frequency identification (RFID) scanner system sensor suite.
  As lead power distribution lead, he will be required to oversee work involved with the design, implementation, and deployment of power conditioning and distribution to the entire mobile system.
  He has been assigned the task of backup lead engineer for vehicle mechanics and structure to assist Brady Boone if so determined by team lead.
  In addition, he will also serve as the Financial Officer in charge of managing all project fund appropriations and expenditures.

- *Chasen Peters:*
  Will serve as lead engineer for the **onboard processing unit and motor control.**
  As lead onboard processing unit engineer, he will be required to oversee work involved with design, implementation, and integration of the onboard central processing unit. This includes, but is not limited to, ensuring sensor communications between sensor arrays and onboard computer as well as wireless data communication to and from command center.
  As lead motor control engineer, he will be required to oversee work involved with design, implementation, and integration of the motor control interface which will drive mobile system.

He has been assigned the task of backup lead engineer for sensors hardware (RFID) and power distribution to assist Paul Naud if so determined by team lead.
In addition, he will also serve as the Team Lead. This includes, but is not limited to, managing all project lead engineers and team members, upholding project standards and policies, and ensuring all tasks are accomplished to meet the goal set forth by the mission statement.

CODE OF CONDUCT

MEETINGS

The team as a whole must meet at least once a week for a stand-up and status meeting. The weekly meeting time will be determined at the end of the previous weeks' meeting, starting on Monday, March 21, 2011. If there is an unforeseen schedule conflict the group should be notified with a 24 hours advanced notice for rescheduling. The agenda for a meeting shall be drafted and sent out by the Documentation Administrator 12 hours before the scheduled meeting time.
Any meetings that would include external attendees such as professors, clients, and/or sponsors, require all team members to arrive a minimum of 30 minutes before the scheduled meeting time.

TIME COMMITMENTS

Each team member is required to work at least 30 hours per calendar week but no more than 60 hours. Inability to meet the minimum hours required per week must be discussed with Team Lead. In addition, any overtime must be pre-approved by Team Lead.
Team members must fulfill their required task within the allotted time. If the deadline cannot be met, the team member must provide adequate justification one day in advance of current deadline, along with a new deadline to the task in question.
Team members arriving late to a meeting must provide justification for tardiness to Team Lead.
Team members that miss or are tardy to a meeting without prior notice are responsible for attaining information from all topics discussed during the missed meeting.

TEAM INTERACTION

When addressing other members of the team, members must interact in a professional manor.
There will be at least one mandatory team building social outing every two calendar weeks. No project work may be discussed on the outings as they will serve as a mental and physical break from daily work. Every team member must attempt to get a minimum of six hours sleep every day to promote efficient and high work productivity.
Any member that visits an In-N-Out is required to purchase items for all team members, and will be reimbursed upon next lab session.

BUDGET CONTROL

The Financial Officer will have control over all team funds and is in charge of dispensing those funds as required and approved per the guidelines list below.
Every expenditure less than $50.00 must be agreed upon by the financial officer, Team Lead, and member attempting to purchase.
Every expenditure greater than $50.00 must be agreed upon by the team as a whole.

DOCUMENTATION MANAGEMENT

Any formal documentation submitted must be added to version control and approved by the Documentation Administrator.
All lab notebooks must be up to date and written in ink. Upon every entry, members must sign and date page(s) worked upon.
Each member is responsible for maintaining a lab binder for the documentation of data sheets and application notes as required for their assigned tasks. Each entry must be indexed and dated for clarity and understanding.

DECISION MAKING AND DISPUTE RESOLUTION

Any dispute that affects the overall project timeline must be discussed between all members of the team. If a decision cannot be met by a 4 out of 5 majority vote, then a third party representative (professor and/or teaching assistant), must be requested to mediate the dispute and offer possible conflict resolutions. If a conflict still remains after the third party intervention, the Team Lead must make a final decision.

TERMINATION POLICY

Should a team member consistently cause delays, under perform, or violates agreements drafted in this charter, the team member is subject to termination from the team. The team member in question must first be approached by at least two other members of the team to discuss the issues at hand. If no improvements are evident, then a third party representative will be asked to mediate the situation. If progress is still not made, unanimous vote from remaining team members will result in termination of this member's position within the team.

ADDENDUMS AND CHANGES TO CHARTER

By unanimous team vote, the team may correct, change, or add sections to this charter at any given time.

AGREEMENT

By providing your signature below you are agreeing to abide by the policies set forth in this charter written for the Autonomous Mobile Porter System project, Jack Baskin School of Engineering Senior Design Winter/Spring 2011.

_____

BRADY BOONE **DATE**

_____

ALEJANDRO CERDA **DATE**

_____

NOLAN LAU **DATE**

_____

PAUL NAUD **DATE**

_____

CHASEN PETERS **DATE**